CANADIAN THESES ON MICROFICHE

THÈSES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Canada

NL 339 (r. 88/09)

National Library of Canada / Bibliothèque nationale du Canada

Ottawa, Canada
K1A 0N4

(81)

TC –

ISBN 0-315-21275-6

CANADIAN THESES ON MICROFICHE SERVICE – SERVICE DES THÈSES CANADIENNES SUR MICROFICHE

PERMISION TO MICROFILM – AUTORISATION DE MICROFILMER

• Please print or type – Écrire en lettres moulées ou dactylographier

AUTHOR – AUTEUR

Full Name of Author – Nom complet de l'auteur

MOHAN PALAT

Date of Birth – Date de naissance
22-12-1957

Canadian Citizen – Citoyen canadien
☐ Yes / Oui ☑ No / Non

Country of Birth – Lieu de naissance
INDIA

Permanent Address – Résidence fixe
KALPANA,
KRISHNASWAMY NAGAR,
COIMBATORE-641045. TN. INDIA

THESIS – THÈSE

Title of Thesis – Titre de la thèse

DATA BASE MANAGEMENT FOR GEO-DATA

Degree for which thesis was presented
Grade pour lequel cette thèse fut présentée
MSC

Year the degree conferred
Année d'obtention de ce grade
1985

University – Université
UNIVERSITY OF ALBERTA

Name of Supervisor – Nom du directeur de thèse
WAYNE A. DAVIS

AUTHORIZATION – AUTORISATION

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to **microfilm** this thesis and to **lend** or **sell copies of the film.**

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de **microfilmer** cette thèse et de **prêter** ou de **vendre des exemplaires du film.**

L'auteur se réserve les autres droits de publication, ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

ATTACH FORM TO THESIS – VEUILLEZ JOINDRE CE FORMULAIRE À LA THÈSE

Signature
Mohan Palat

Date
24-4-1985

NL-91 (r. 84/03)

Canada

THE UNIVERSITY OF ALBERTA

Data Base Management for Geo-Data

by

Mohan Palat

(C)

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF Master of Science

Department of Computing Science

EDMONTON, ALBERTA

Spring 1985

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR Mohan Palat

TITLE OF THESIS Data Base Management for Geo-Data

DEGREE FOR WHICH THESIS WAS PRESENTED Master of Science

YEAR THIS DEGREE GRANTED Spring 1985

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) Mohan Palat .....

PERMANENT ADDRESS:

KALPANA

KRISHNASWAMY NAGAR

COIMBATORE - 641045

T.N. INDIA

DATED 24th April 1985

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled Data Base Management for Geo-Data submitted by Mohan Palat in partial fulfilment of the requirements for the degree of Master of Science.

W. A. Davis
.....
Supervisor

.....

.....

.....

.....

Date 18 APR 85

ABSTRACT

This thesis is concerned with data base management systems for geographic applications. A survey is presented of the various geographic information systems that are based on data base concepts. The strengths and weaknesses of each are identified and different systems are compared for overall performance. An analysis is then made of the three conventional data models for their usefulness in geographic applications. An implementation of a spatial data base on the INGRES DBMS running under UNIX is presented. The implementation is also shown of an associated query language developed for this system. A hybrid data model which evolved from the analysis of the three conventional models and was used in the implementation is also discussed. Finally, the extension of geographic data base management systems to a distributed environment is presented and the various issues involved in such an extension are explored.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor, Dr. W. A. Davis, for his guidance, encouragement and constructive criticism throughout the course of this research. His comments and suggestions throughout this work contributed greatly to the contents and style of this thesis.

I am also grateful to the members of my examining committee, Dr. J. Mowchenko, Dr. Tamer Ozsu, Dr. M. Aoki and Dr. S. Cabay for their useful suggestions.

I also wish to thank the Department of Computing Science for their financial assistance.

Thanks are also due to my friends for the various forms of assistance that they have given me during my stay in Edmonton.

Finally, I am indebted to the members of my family, especially Amma and Achan, for their continued support and encouragement throughout my life.

Table of Contents

Chapter — Page

1. INTRODUCTION 1

   1.1 Basic Concepts. 2

   1.2 Thesis Organization. 5

2. COMPARISON OF DATA MODELS FOR GEOGRAPHIC APPLICATIONS 7

   2.1 Specific DB Requirements for Geographic Applications. 7

   2.2 The three data models. 10

      2.2.1 Introduction. 10

      2.2.2 Network Model. 11

      2.2.3 Hierarchical Model. 13

      2.2.4 Relational Model. 13

   2.3 Comparison of the three models. 16

      2.3.1 Data Representation. 16

      2.3.2 Information Retrieval. 18

      2.3.3 Other General Issues. 20

      2.3.4 Other Geographic Issues. 21

3. GEOGRAPHIC INFORMATION SYSTEMS - A SURVEY 24

   3.1 Evaluation Criteria 25

   3.2 Specific Systems. 26

      3.2.1 ALIS: Areal Land Information System. 27

      3.2.2 ADAPT: Areal Design and Planning Tool. 30

      3.2.3 LUIS: Land Use Information System. 32

      3.2.4 AQL: A Query Language 34

      3.2.5 ADM: Aggregate Data Manager. 37

3.2.6 GADS: Geo-data Analysis and Display System 39

3.2.7 GDB: A Geographical Data Base 41

3.2.8 GEO-QUEL: A Special Purpose Geo-data System ...44

3.2.9 MAPS: Map Assisted Photo Interpretation System 46

3.2.10 ARC/INFO: Computer Mapping and Geographic Information System 48

3.2.11 IDMS: Integrated Data Management System 51

3.3 Conclusions. 53

4. THE INGRES DBMS FOR GEOGRAPHIC DATA MANAGEMENT 56

4.1 The INGRES Process Structure. 57

4.2 The INGRES File Structure. 58

4.3 INGRES for Geographic Applications. 59

4.4 Summary 63

5. DESIGN AND IMPLEMENTATION OF A GEOGRAPHIC INFORMATION RETRIEVAL SYSTEM 64

5.1 Scope of the System. 64

5.2 System Architecture. 65

5.2.1 Introduction. 65

5.2.2 File System Kernel Module. 67

5.2.3 UNIX File System Module. 68

5.2.4 Data Base Module. 69

5.2.4.1 System Requirements. 70

5.2.4.2 Proposed Hybrid Data Structure. 71

5.2.4.3 Data Base Query Language. 74

5.2.4.4 Query Language Considerations. 77

5.2.4.5 Advantages of a Front-End Approach 80

5.2.4.6 Disadvantages of a Front-End Approach .82
5.2.4.7 Performance Considerations. 83
5.3 Conclusions. 84

6. DISTRIBUTED GEOGRAPHIC DATA BASES 85
6.1 Motivation for distribution. 86
6.2 Distributed Data Base Architectures. 91
6.2.1 Centralized Data Base, Distributed Users 91
6.2.2 Partitioned Data Base 92
6.2.3 Replicated Data Base 93
6.3 Distributed DBMS Functions 93
6.4 A Distributed Geographic Information Retrieval System 95
6.4.1 Data Distribution Strategy 96
6.4.2 Data Base Allocation Strategy 97
6.4.3 The Distributed Environment 100
6.4.4 Query Processing Strategy 102
6.5 General Distributed Geographic Data Bases. 105
6.6 Conclusions 106

7. CONCLUSIONS 111
7.1 Further Research. 113
REFERENCES 115

APPENDIX I 119

APPENDIX II 135

APPENDIX III 136

APPENDIX IV 137

CHAPTER 1

INTRODUCTION

This thesis is concerned with the issue of data base management systems for geographic data. The field of geography has come a long way since the days when mapmaking was the work of artists. Several important innovations, of which automation is one of the most recent, have revolutionized this field. Of these innovations, electronic data processing and communications have had a profound effect, changing both its concept and its process. The many advances in electronic data processing over the years have moved automated computer geographic data processing from the stage of exploratory development to the more mature arena of realizable results. One such advance is the concept of data base and data base management systems. This thesis deals with some of the issues involved with the application of this concept to geographic information systems.

Over the years, several geographic information systems have been designed and implemented [22]. However, most of them are special purpose systems that are not based on the concept of data bases. Such systems have several drawbacks that resulted in unsatisfactory performance and limited their usefulness. However, the application of data base techniques to this field have prompted a re-thinking of the

performance and usefulness of geographic information systems. There has not been a great deal of study done about the various issues involved. This thesis attempts to fill part of this void.

There have been several approaches to designing a geographic data base system [21,22]. One of the most cost-effective approaches is a *front-end* one [27]. However, to make this approach feasible, it is very important to have efficient data structures. An attempt has been made in this thesis to construct one such structure, for a *front-end* data base system.

In practical applications, most of the geographic data sites are geographically distributed. This makes the study of distributed geographic data base systems very crucial to future efforts to make such sytems practically feasible. In this thesis, the issue of distributed geographic data bases has been touched upon.

In summary, this thesis seeks to provide a better understanding of the issues involved in the area of geographic data base systems.

1.1 Basic Concepts.

Geographical entities can be divided into three data types: *point*, *line* and *region* [21]. A line can be represented by a succession of points and a region can be represented either by a succession of points or a closed set of lines.

Using the above three data types, six types of binary relations among entities have been defined [21]. They are: *Point-Point Relations*, *Point-Line Relations*, *Point-Region Relations*, *Line-Line Relations*, *Line-Region Relations* and *Region-Region Relations*. On these basic types of relations, several geometric operations can be defined. They include: (1) *containment*, which determines whether any entity is contained inside another entity, (2) *adjacency*, which specifies whether two given entities are adjacent to one another, in location, (3) *intersection*, which determines whether any two given entities intersect each other, and (4) *closest point*, which basically defines the point closest to a given entity. In the above operations, an entity may either be a point, a line or a region.

The information attached to geographic entities can be classified as *geometric* and *non-geometric* [21]. Geometric information generally specifies the location and shape of the entity, while non-geometric information refers to descriptive information about the entity. This is illustrated in Fig. 1.1.

Based on this classification, data bases for geographic applications have been divided into two categories: *cartographic data bases*, containing geometric data and *statistical data bases*, containing non-geometric data.

Geometric data may be represented in two formats: *vector* and *raster* [22]. The internal data organizations corresponding to these two formats are called a *linked*

Entity :	FORT McLEOD
Feature :	Historical site
Source :	GAZ
Description :	Situated in Alberta
Latitude1 :	47 N
Latitude2 :	49 N
Longitude1 :	112 W
Longitude2 :	113 W

The above information can be split up into geometric and non-geometric information, as shown below.

Non-geometric information:

Feature :	Historical site
Source :	GAZ
Description :	Situated in Alberta

Geometric information:

Latitude1 :	47 N
Latitude2 :	49 N
Longitude1 :	112 W
Longitude2 :	113 W

Fig. 1.1. An example of geometric and non-geometric information in a geographic entity.

organization and a *cellular* organization, respectively [22]. In the latter, the extent under consideration is divided into a matrix of cells of uniform size, and computer storage is set aside for each cell. In the linked organization, point entities are described directly by their coordinates, line entities by a chain of uniformly or nonuniformly spaced coordinates and regions by closed curves.

Geographic data is manipulated by eight basic operations recognized by Page and Wilson [33]: Accessing,

Inserting, Deleting, Searching, Sorting, Copying, Combining and Separating. These operations are the same as that for other types of data.

In the rest of this thesis, these basic concepts have been used freely, as and when found necessary. Their definitions and meanings remain the same as stated in this section, unless revised definitions are explicitly provided.

1.2 Thesis Organization.

The thesis has been divided into seven chapters. In Chapter 2, a study of the three conventional models - relational, network and hierarchical, for geographic applications, is made. The advantages and drawbacks of the three models for the storage and manipulation of geographic data are also highlighted.

The next chapter is a survey of some geographic information systems. This survey is intended to provide an overview of recent developments in this area.

In Chapter 4, the capabilities and drawbacks of INGRES (Interactive Graphics and Retrieval System) for geographic data is discussed.

In Chapter 5, the details regarding the design and implementation of a specific geographic data base on UNIX' is elaborated.

The next chapter deals with the issue of distributed geographic data bases. The motivation for distribution is

---

'UNIX is a trademark of AT&T Bells Labs

enumerated and the various issues involved in the extension of a data base to a distributed environment are studied.

Finally, in the concluding chapter, the various results are summarized and areas of further research in this field are identified.

CHAPTER 2

COMPARISON OF DATA MODELS FOR GEOGRAPHIC APPLICATIONS

In this chapter, the three conventional data models are compared for their usefulness in geographic applications. The objective of this comparison-study is not to make any definite conclusions as to which model is most suitable for geographic applications, but to highlight the different strengths and weaknesses of the three models in such an environment.

2.1 Specific DB Requirements for Geographic Applications.

Geographic applications differ from conventional business applications in three basic areas: the type of data that they deal with, the volume of data that they deal with, and the type of operations that they require. Geographic data have certain specific data base requirements that are not part of conventional data base systems [31]. These include:

1. The data structure that the model allows should ideally be suitable for geographic operations like containment, adjacency, and intersection. In some cases, the number of object coordinates can be quite high which can result in a large number of pointers. This can in turn dramatically increase the overhead cost. The data

structure must therefore economize on the use of pointers.

2. Geographic data is basically of two types: spatial data and aspatial data (also referred to as geometric data and non-geometric data respectively). Many queries require the manipulation of both spatial and aspatial data. The following query is an example:
   Find all roads in Edmonton which are more than 3 feet wide.
   The organization and manipulation of the two kinds of data require different facilities. However, the spatial and aspatial data must be related in some way to facilitate efficient query processing.
3. The user should be able to access data by name, so that programs are independent of the data that they access. In other words, the data should be self-describing.
4. The data model should support the use of expressions as data values. In many geographic applications, entities are related to one another and when one entity changes its attribute values the related entities must also change their corresponding attribute values.
5. The data model should support ordered files in which data is organized in some sorted order. Ordered files are very crucial in applications where, for example, a sequence of (x,y) coordinates represents a geometric figure.
6. Geographic applications require a different set of data

operations as compared to conventional business applications. Apart from conventional operations like selection, deletion, insertion and search, geographic applications require operations like containment, adjacency and intersection.

7. Unlike conventional data, geographic entities are usually distinct in that they are usually related only by geographic location. For example, City X is related to Country Y only if X is in Y. Correlation of data in a commercial data base is much more intricate.
8. A geographic entity (e.g., a map) is usually continuous whereas commercial data has no continuous quality. Therefore, it is not always convenient to split up and store entities of the former type in a discrete format, as in the latter type.
9. Again, the output of a commercial data base is usually in alphanumeric printed form whereas geographic data bases normally support graphic output.

In general, conventional data models are not always suitable for geographic applications because

1. they do not support the type of operations that are required for this type of applications, and
2. they are not ideal for the storage and manipulation of spatial (or geometric) data.

2.2 The three data models.

In this section a brief description of the three data models relational, hierarchical and network is provided. This serves as an introduction to the discussion, in the next section, on the relative strengths and weaknesses of the three models.

2.2.1 Introduction.

A data model basically consists of a mathematical notation for expressing data and relationships and a set of operations for the manipulation of the data [29]. There are three important data models used in the bulk of conventional data base management systems. They are the relational, hierarchical and network models [29]. The network model is based on a single source - the proposals put forth by the Data Base Task Group (DBTG) of the Conference on Data Systems Languages (CODASYL)[7]. The hierarchical model is a special case of the DBTG model. The relational model, on the other hand, has no such guidelines for its development. This means that the relational implementors have greater freedom in making design choices.

Fig. 2.1 is a simple data base that contains geographic information about the relationship between a city, its streets, and its roads. In the figure, rectangles represent real-world entities, diamonds represent relationships among them and names connected to rectangles by straight lines represent attributes of the entities. In the succeeding

sections, this example will be used to illustrate the representation of information in the three data models - network, hierarchical and relational.

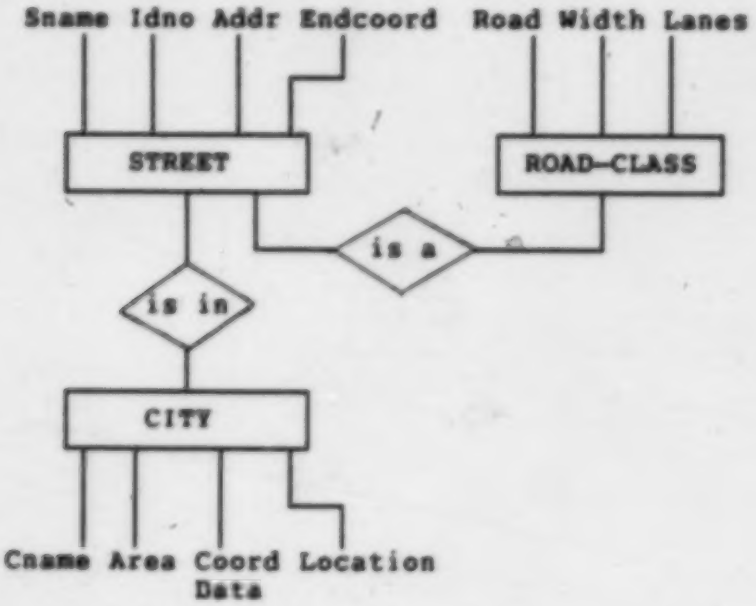


Fig. 2.1. An example of a simple geographic data base

2.2.2 Network Model.

The network approach is characterized by the CODASYL DBTG proposals [7] and consists of a collection of one or more record types corresponding to types of real-world entities. Each entity type has characteristics or attributes that are described by the various fields within the record. Each record type is completely defined using a data description language (DDL). The data is manipulated using a data manipulation language (DML).

Any node in the network (corresponding to entities) can be related to any other node. However, all the relationships among the nodes are restricted to be binary, many-to-one [29]. This restriction of many-to-one relationships allows the use of a simple directed graph model for data. However, arbitrary relationships can be represented as two or more many-to-one relationships with the use of dummy record types [29].

Two types of operations can be performed on a network data base. One is the selection of record types and the other is navigation along the links in the data base which is similar to the join operation in relational data bases (to be explained in a later section).

In the network model, the data base in Fig. 2.1 can be represented by Fig. 2.2.

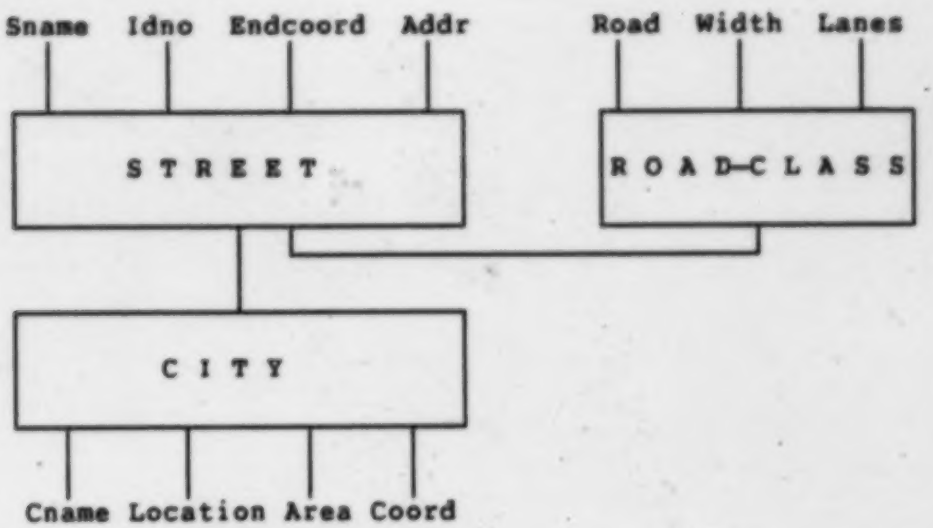


Fig. 2.2. Network representation of the data base

2.2.3 Hierarchical Model.

A hierarchy is simply a network that is a collection of trees in which all the links point in the direction from child to parent. Therefore, a hierarchical data base [29] is a collection of disjoint trees with records as nodes. Hierarchical structures, or trees, exhibit the following characteristics: it is called the root.

1. Each node can have zero, one, or many nodes related to it on a lower level; they are called children.
2. Each node, except for the root, can be related to only one node at a higher level, called the parent.
3. Nodes which have no children are called leaves.

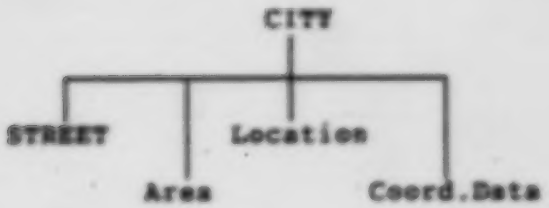
The above characteristics confine the relationships among enitities to one-to-many. However, many-to-many relationships can be implemented by using the concept of virtual records [29].

In the hierarchical model, the data base in Fig. 2.1 can be represented by the three trees in Fig. 2.3. The trees are formulated using the set of rules specified to organize a data base in the hierarchical model. In the figure, the third tree consists of only one node, corresponding to a single record type. Therefore, it is a tree in a trivial sense only.

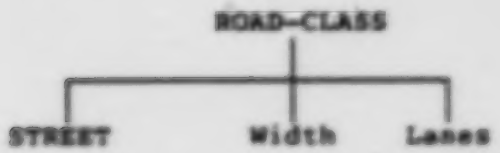
2.2.4 Relational Model.

A relational data model [29] is one in which the data is represented in the form of relations (or tables). Each

Tree 1:



Tree 2:



Tree 3: (SName, Idno, Endcoord, Addr)

Fig. 2.3. Hierarchical representation of the data base

relation has a fixed number of attributes (or columns) corresponding to the attributes of the real-world entity that it represents. A relationship among entities is also represented by a relation. Each row, called a tuple, describes a specific enitity or specific aspects of an enitity.

The mathematical concept underlying the relational model is the set-theoretic relation, which is a subset of the cartesian product of a list of domains (set of values). Languages for expressing queries about relations are based on two mathematical concepts - relational algebra and predicate calculus. In the former, queries are expressed by

applying specialized operators to relations and in the latter, queries specify a predicate that the resultant tuples must satisfy.

A variety of operations have been defined for the relational model. These include operations like union, set difference, cartesian product, selection, projection, join and intersection [29]. The model presents a flexible data structure, since new relations can be created from existing relations merely by using some of the above mentioned operations.

Relations have three important characteristics. First, there cannot be identical rows and hence each tuple in the relation must have a unique or primary key. Second, the order of the rows is insignificant; that is, a sorted order is not necessarily maintained. Finally, every column consists of a non-decomposable value.

The geographic data base example of Fig. 2.1 can be represented in the relational model by the following:

STREET (SName, Idno, Endcoord, Addr)

CITY (CName, Location, Area, Coord.)

ROAD-CLASS (Road, Width, Lanes)

IS-IN (Id.no, CName)

IS-A (SName, Road)

The above representation is not a normalized one. A normalized data base is basically one in which most of the

problems of redundancy and anomalies are eliminated.

So far, in this chapter, the three conventional data models have been briefly described. In the next section these three models are compared for their advantages and drawbacks.

2.3 Comparison of the three models.

Both the hierarchical and network models are, in general, based on the CODASYL DBTG proposals. In fact, the hierarchical model can be considered as a network that is a forest (collection of trees). Therefore some of the features of the network model are inherent in the hierarchical model which avoids the need to consider them separately in the comparison of those features that are common to both [20].

2.3.1 Data Representation.

The relational model can be considered as a collection of tables, each of which has one or more columns, that correspond to attributes. Tuples in a relation correspond to entities, relations correspond to entity sets and the attributes of a relation correspond to features of the corresponding entity set. Interestingly, both the entities and the relationships among entities are represented by tuples in a relation.

The DBTG models basically consist of record types corresponding to entity sets and links corresponding to relationships among them. The data items correspond to

attributes in a DBTG model. The term DBTG model, in this thesis, refers to the models based on the CODASYL DBTG proposals, namely the network model and the hierarchical model.

The values of the attributes in a relation in the relational model are usually simple. However, in a DBTG model they may be complex. For example, in a DBTG model a repeating group of attributes can be defined by the user. Again, the relations in a relational model usually need to be normalized, but this is not a requirement in DBTG models. Normalization, however, results in simpler relations that generally reduce redundancy and update problems.
The concept of keys in a relation imposes a constraint that no two tuples can be identical. In a DBTG model, this decision is usually left to the DBA or the user. The two approaches also differ in the way relationships are represented in the data base. The DBTG sets, corresponding to relationships, are a very useful mechanism for building complex data structures, in a DBTG network model. Moreover, the DBTG sets also facilitate network traversal, i.e., navigation, and establishment of access paths. This allows the search/retrieval routines to traverse through different record types to locate the desired information. This concept is not included in the relational model. The DBTG DDL, however, is in many cases unusually complex and incomplete in its data description as opposed to the relational approach which is fairly straightforward and simple.

2.3.2 Information Retrieval.

Information is retrieved from the data base with the help of data manipulation languages. The data manipulation facility may be imbedded in the host programming language or it may be a special language.

As mentioned in an earlier section, relational manipulation languages are based on two mathematical concepts: relational algebra and relational calculus. The data manipulation language, as specified in the DBTG proposal, is an extension of the programming language COBOL and depends on the host language to control program logic. The two approaches are distinctly different. One is based on set theoretic operations while the other is based on path traversal techniques.

Geographic applications usually require a data manipulation language to operate in an interactive, real-time environment. However, the data manipulation facility in a DBTG model is usually embedded in the host programming language, and hence, is not very suitable for an interactive environment. The relational languages, on the other hand, are designed for such an environment.

The *completeness* of a data manipulation language refers to its ability to provide a set of query operations that is complete, in the sense that it can handle any type of query. Relational languages have been proved to more "complete" than the network and hierarchical languages [3,7,8,20]. Relational languages operate with sets while the DBTG DMLs

operate on a single record at a time. Hence it is often necessary to include control statements in a query in DBTG languages. This makes them more verbose than relational query languages which have a greater degree of conciseness.

In geographic applications, most users are non-programmers. Thus the ease of use and hence the language level and language complexity are important issues in any comparison. Higher level languages are more user-friendly in the sense that they allow the user to specify what is to be done without specifying how it is to be accomplished. Stonebraker and Held [11] have concluded that relational query languages are generally higher level than the DBTG DMLs. The latter, however, is useful in applications which require an access level close to the storage structure.

Language complexity basically refers to the complexity of the formulation of queries. In the relational approach, both the entities and relationships are represented by tuples, i.e., records, whereas in the DBTG approach, they are represented as records and DBTG sets respectively. Consequently, the relational approach requires a smaller collection of operators to retrieve information as compared to the DBTG approach.

Besides the issues discussed so far, there are some other comparison issues that highlight the differences between the three models. The next section deals with these issues.

2.3.3 Other General Issues.

In the DBTG model, concurrency protection is primarily the responsibility of the user in that it is specified within the DML unlike the relational approach. Waghorn [30] has concluded that, in general, concurrency protection is inadequate in a DBTG model.

The issue of data independence can be divided into two categories: physical data independence and logical data independence. The former is guaranteed in the relational model because neither the data definition language nor the data manipulation language contain any reference to either the physical storage of data or the access mechanism. This is not true in the DBTG model, because the DDL contains facilities to control certain aspects of the storage structure. As far as logical data independence is concerned, the problems associated with the addition and deletion of records and data items and the re-arranging of data are present in both the DBTG and the relational models [20], although it is more complex in the DBTG model than in the relational model. In the DBTG approach, the user must be concerned with changes in access paths and relationships among entities. In the relational approach, there is a possibility that the relation would be un-normalized with the addition of new data.

2.3.4 Other Geographic Issues.

Geographic applications require geometric operations such as containment, adjacency, intersection and closest point. In the relational model, these are translated to operations such as union, intersection, join, projection and selection. In the network model, there are two basic types of operation: selection and navigation. In the hierarchical model, the basic operation is a tree-traversal. These data base operations however are not good primitives for geographic operations which are inherently geometric. For example, a line-region intersection query like,

Does river X cross forest Y?

is reduced to searching for segments with the same pixel position.

Again, the relational representation of vector data as attribute-value tuples requires the primary key attributes to be duplicated in each relation since the relational model does not allow multiple valued sets as a primitive attribute. This is not true with DBTG models like network and hierarchical where the DDL can define a repeating group of attributes, which may occur more than once for each occurrence of the record type. The following is an example of a definition for a repeating group of attributes in a DBTG model:

Records

Record name is City

Name character(x)

Roads occur multiple times

etc...

Geographic data bases are generally very large. In such an environment, logical partitioning of the data base must be done to avoid performance degradation for spatial search operations. This is difficult to achieve in relational systems because the relational model restricts itself to homogeneous sets. In a DBTG model, the DML operates with one record at a time. Because of their hierarchical tree-like structure, limited partitioning is possible for search operations on spatial data.

In this chapter, the three conventional data models - relational, hierarchical and network, have been analysed for their strengths and weaknesses, for geographic applications. However, no attempt has been made to make a definite conclusion as to which model is best suited for such applications.

SUMMARY OF THE GEOGRAPHIC CAPABILITIES OF THE THREE DATA MODELS

Issue	Relational	Network	Hierarchical
Geometric operations	Fair. Translated to relational operations	Poor. Only selection and navigation operations	Poor. Only tree-traversal operation
Geographic data representation	Poor.	Limited capability	Limited capability
Large volumes of data	Deals only with homogeneous sets	Limited logical partitioning	Limited logical partitioning
Geographic query facilities	High-level Easy to use	Low-level More complex	Low-level More complex
Data representation	More natural and simpler	Relatively complex	Relatively complex
Data independence	Good	Poor	Poor

CHAPTER 3

GEOGRAPHIC INFORMATION SYSTEMS - A SURVEY

In this chapter, a survey of the different types of geographic information systems is presented. Special emphasis is placed on geographic information systems that are based on data base concepts.

Until recently, the external organization of the data in geographic information systems was as *data banks* [22]. The data banks are "separation" oriented in the sense that the information is segregated either on the basis of type of entity or on the basis of subextent or both. In such systems, the files contain data in simple sequential order with little or no cross-referencing. However, recently, many geographic information systems that organized the data as data bases have been designed and implemented [22].

The primary emphasis in this thesis is on geographic information systems that are based on data base concepts. From this point of view, geographic information systems can be broadly classified into two types: those that are based on data base concepts and those that are not [see Fig.3.1]. The former can be further classified into two types: special purpose systems that are designed and implemented for specific applications, and general purpose systems that can be used for a variety of geographic applications. The

general purpose systems can be either specially designed systems or extensions (add-on) to existing general data base management systems.

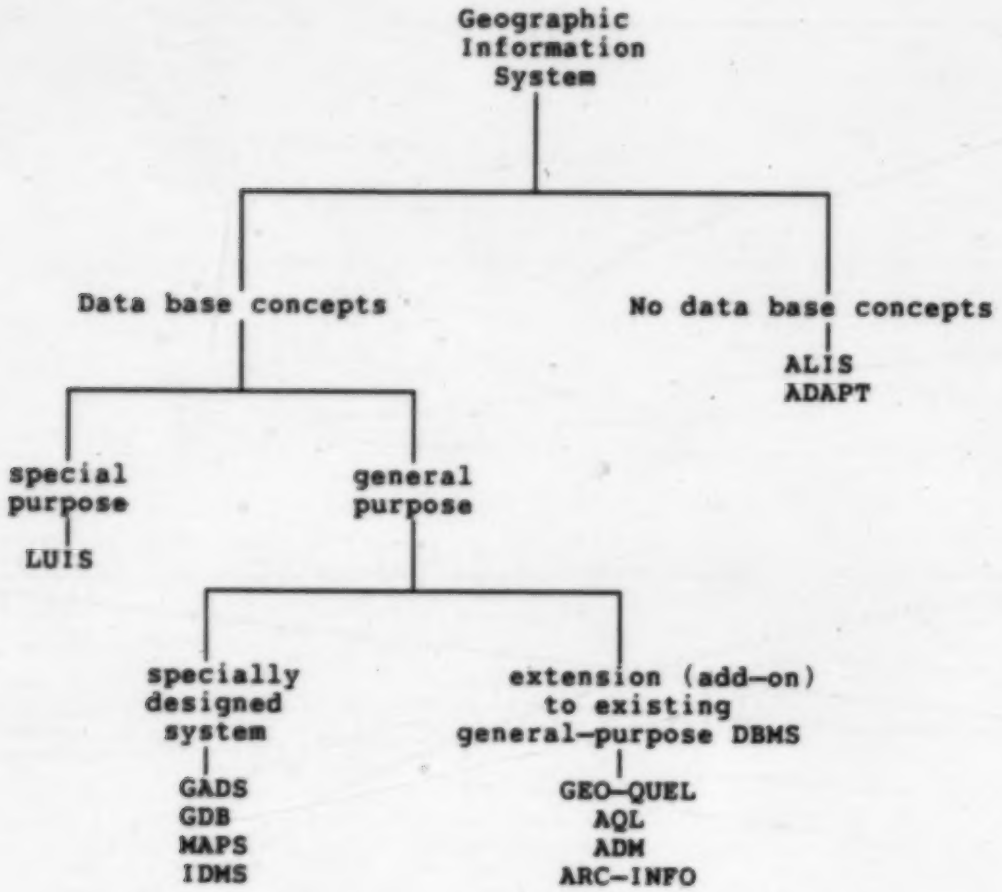


Fig. 3.1. Classification of Geographic Information Systems

3.1 Evaluation Criteria

In this section, some of the criteria used to evaluate the different types of geographic information systems is listed. They include:

- The capabilities of the system to relate image-based

features to a map coordinate system.

- The facilities to incrementally update feature descriptions based on updates to the source image.
- The portability or machine independence of the system.
- The display capabilities supported by the system.
- The facilities for the representation of complex spatial relationships.
- The recognition and handling of inconsistencies in the data.
- The ability to accept information from a variety of input sources.
- The synthesis tasks supported by the system.
- The query language capabilities of the system.
- The spatial relationships computed by the system in executing the queries.
- The completeness of the implementation of the system.

The criteria listed in this section cover all the major factors of comparison of different systems. These factors form the basis of comparison of some specific geographic information systems, which are presented in the next section.

3.2 Specific Systems.

In this section, brief descriptions of some selected systems which cover all the classes of geographic information systems mentioned earlier is presented. The information about each system was obtained mainly from the

literature [37,16,19,1,28,17,13,27,18,35,36]. While the selected systems cover all the types of systems, the emphasis is on those that are based on data base concepts. The primary objective of the analytical sketches is not to highlight specific systems, but to present the different *types* of systems that are in use in geographic applications.

3.2.1 ALIS: Areal Land Information System.

ALIS [37] is a geographical information system for urban research, developed by the University of Tokyo and five other universities in Japan. The work was started in 1976 and is at present complete in its implementation. ALIS is basically used to present up-to-date information in the form of interactive maps to urban planners.

ALIS consists of several subsystems: data collection systems, thematic data bases, cartographic data bases, a data base handler, statistical/spatial analysers, simulators and display systems. The software for all these systems were written in Fortran IV, to enhance portability. These subsystems are generally machine-independent and can be moved from one computer/output device to another with ease. In fact, the system can be adapted to most raster output devices through small interface/conversion programs.

The design of the system is based on sharing the workload between mainframe host computers and intelligent workstations. The intelligent workstations make it possible to display or print hard copy in real time or off-line. This

system is basically a file system and is not based on any data base model. There are two classes of data: *thematic data* which is either 2-D matrix coded data or one dimensional data and *cartographic data* containing both raster and vector data. The thematic data characteristics are summarized in directory files.

The manipulation of the data is done through the *data base handler*. Most of the input is done through the data collection systems. An address coordinate conversion system converts a physical location into latitude & longitude, to produce grid data. A set of over 80 commands that can be used interactively constitute the query language. However, this is basically only a set of functions written in Fortran and not really a query language.

The display facilities are machine independent and the system can be adapted to most raster output devices through small interface programs that convert the standard ALIS raster format to that of the device being used. The system has interactive high resolution raster display facilities. The graphics is raster based and all vector commands are converted to raster format. The system requires at least 3200K of memory for object code of its programs and the intelligent workstations which share data, each require at least 48K of memory.

Advantages:

- The system has good graphics display capabilities.
- The system can accept input from a variety of sources.

- The subsystems of ALIS are generally machine independent.
- The system has an address conversion system which helps relate image-based features to a map coordinate system. Map coordinate system refers to the coordinate system (including reference points etc.) that is used in a particular map while image-based features refer to the features (like, river, lake) that are present in any specific image (captured data). This mapping feature, however, is restrictive and specialized, and cannot handle general cases.

Drawbacks:

- Since the system is a file system, complex spatial relationships cannot be represented in the data base. Data is stored in the file system in either vector or raster format. The manipulation of the data is done by the data handler.
- The system has no facility to check inconsistencies and redundancies in the data.
- The system has limited capabilities to perform spatial operations on the data.
- Its command language set is not really a query language but a set of Fortran routines which imposes restrictions in its use.

Summary:

ALIS is based on file system without any data base concepts. Therefore, it has most of the disadvantages

associated with the lack of a data base environment. The system has good graphics capabilities and is machine independent. In the environment for which it was designed, ALIS has proved to be very useful, but in others it places many restrictions on the users.

3.2.2 ADAPT: Areal Design and Planning Tool.

ADAPT [16] is a geographic information system designed to support analytical modeling, combining spatial data management with analytical modeling capabilities. The system was developed by W.E.Gates & Associates and work on it was started in 1972. The system is complete in its implementation and is written in Fortran. ADAPT is different in concept and orientation from most geographic information systems in that the representation of spatial data is a model and that the measure of performance is not the ease of manipulation, but whether or not improved insights, better decisions, or superior designs can be achieved as compared to conventional methods. Hence, it can be best described as a *geo-based modeling system*. The fundamental data storage mechanism in ADAPT is a cell-based triangular irregular network(TIN). In TIN, surfaces are represented by triangular facets; the number of facets can be increased locally to provide more fidelity of representation when required. This is particularly suitable for modeling terrain. In the actual data structure, the latitude, longitude and elevation of each vertex is recorded; this explicitly defines each

triangle as a plane in space.

Input data is required to be organized as cell-based triangular networks with attributes assigned to cells, lines and points within the network. X-Y coordinates and statistical attribute data are related through the triangular cells and have attributes associated with them. Files are direct access in ADAPT. The display capabilities include pen and printer plot outputs, contour plots, polygon plots and shaded plots. The system does not have any interactive query facility. Each time a display is required, a display program has to be executed.

Advantages:

- Complex spatial features can be modeled in the system by using the triangular data structure.
- Much of the software required for processing the data is of general nature, and can be used to process and display arbitrary point, line or cell data.
- The display capabilities of ADAPT include a variety of printer and plotting devices.

Drawbacks:

- ADAPT is not portable.
- The system has poor query capabilities.
- It has no facility to map image-based features to a map coordinate system. Instead, the input data has to be organized in the required triangular format.
- This approach also introduces redundancy in the data.
- It has no mechanism to check inconsistencies in the

data.

Summary:

In summary, ADAPT is a geo-based system that combines spatial data management with analytical mathematical modeling and is different in orientation from most other geographic information systems. As a result, some of the requirements of a spatial information system have been ignored in the design.

3.2.3 LUIS: Land Use Information System.

LUIS [19] is an interactive graphics system used for data base management in a regional planning environment. It was developed by The Greater Vancouver Regional District(GVRD). Although the system is complete in its implementation, it is being continually expanded and updated to include more utilities and functions. The system was written in BASIC.

LUIS started out as a system for manipulating one specific data base but has now been generalized to handle any data base of geographically-oriented integer information. In LUIS, the geographic information is divided into two elements: the location where the activity is taking place and the description of the activity. These two elements are handled by two sets of file structures: the plot file structure and the data base file structure. A plot file, which is based on the polygon method of plotting, consists of two associated files: the coordinate file, which

contains nothing but strings of coordinate pairs and the pointer file, which contains information about each polygon in the coordinate file. The data base file consists of files of data, organized in the form of relations with fixed length fields/attributes. The data bases contain both statistical and cartographic information.

The data base file structure bears a close resemblance to the relational model of data. The system has been structured in modules linked under the control of a main, supervisory program. This has been done mainly due to memory restrictions. Input data is not restricted to a single format. Small programs translate the input data into the format of the final files of the system. The system supports a large set of interactive query and display commands. The commands are easy to use in that they guide the user through a series of prompts during each interactive session.

Advantages:

- Input data is not restricted to a single format.
- LUIS supports a large set of interactive query and display commands.
- The query language is easy to use.
- LUIS is easy to expand since it is structured in modules.
- The data base file structure allows the search space to be partitioned when computing queries.

Drawbacks:

- The system is not portable, and is designed to run on a

specific machine. This is a major drawback of the system.

- And since it runs on a specific machine, it is also restricted by memory limitations on that machine.
- Several spatial operations like intersection, are not supported at the time of the paper[19]. They are, however, included in future plans. Therefore, the set of dynamic spatial queries is incomplete. A facility to incrementally update feature descriptions is also included in future plans.

Summary:

LUIS is a special purpose system. It is basically used for regional planning applications. One of the major drawbacks of this system is the incompleteness of its set of spatial operators. It has also some advantages like its expandability and its ease of use. In summary, the system is quite useful for specific applications in the environment for which it was designed. However, as a general purpose system, its usefulness is very limited.

3.2.4 AQL: A Query Language

AQL [1] is a data base management system based on the relational model. The query language is based on the high-level language APL. AQL is implemented on the IBM family of machines. Like INGRES (QUEL), AQL is basically a relational data base management system for business applications that has been extended to handle geographic

data. However, at the time of [1] the system was not complete in its implementation.

Data is organized in the form of relations in AQL. Manipulating geographic data in a relational data base requires the system to have both *computational* and *geographical* capabilities. In AQL, maps are divided into square-shaped cells (zones) and each cell (zone) is represented by a tuple which describes its boundaries and gives it a unique identification to what other information may be attached. Boundaries are viewed as a set of curves and these are constructed by using numerical analysis techniques on a corresponding least set of points stored in the data base for each curve. The reference system is specified in terms of the latitude, the longitude and the height. A map is represented by the relation:
MAP (STARTX, STARTY, ENDX, ENDY, F#, LID, LEN)
where STARTX, STARTY are the latitude and longitude of the lower bound of the boundary interval, ENDX,ENDY are the same for the upper bound, F# is the identifier of the relevant parameters of the best-fitting polynomial, LID is the identifier of the line described by the tuple and LEN is the length of the boundary segment.

AQL also introduces the concept of a hierarchical organization of entities (e.g., "city" is part of "province"; "province" is part of "country"). This concept is accomodated by the concept of *level* in a map.

The complex operations on the maps include union and intersection between maps, overlay of maps, coloring of maps etc. These are represented by queries such as ISPARTOF (is part of), BORDEROF (border of a map), DRAW, ISONEOF (is one of), ISIN (is in) etc. in AQL.

AQL does not provide input/output or display capabilities. It is a geographic data base management system (GDBMS) which should be differentiated from a geographic information system that includes a GDBMS, a display system and an input/output system, among others.

Advantages:

- It is flexible in the sense that it allows the user to define functions from pre-existing commands and thereby extend the language.
- Complex spatial operations are handled by the system.
- Inconsistency and concurrency are handled by the standard mechanisms used in most data management systems.
- Dynamic spatial queries are also supported by the system.

Drawbacks:

- AQL is not portable.
- AQL does not provide input/output or display capabilities.

Summary:

AQL has very useful capabilities for the manipulation of spatial data. It is only a DBMS, however, and not a

complete system by itself. Hence it has to be interfaced with some compatible I/O subsystems to make it complete.

3.2.5 ADM: Aggregate Data Manager.

ADM [28] is an interactive image-oriented data base system developed by IBM Japan. It runs under VM/370-CMS and is coded mainly in PL/1, but the image handling portion is coded in assembler mainly because of bit handling requirements. The relational data base management system (DBMS) SYSTEM-R is used as the basis of ADM and SEQUEL, the query language of SYSTEM-R, is used as the data language in ADM. The system was only partially operational at the time of writing [28].

In this system, each *image* is defined as a data-type and therefore, each image is treated as a data element in a relational data base. This approach enables the usage of the relational data language to handle image data in the same way as non-geometric data. Image data is of two types: binary images and gray-tone images. Functionally, the system consists of four subsystems: the interaction subsystem which controls man-machine interaction, the edit/process subsystem which consists of editors for each data type, the workspace subsystem which basically serves as a temporary workspace for data and the data base subsystem which provides an interface to the data and controls access to the data base.

Spatial or geometric data is generally stored separately and spatial data identifiers are stored in

relations in the data base along with conventional non-geometric data to establish the relationships among them. User interface commands are of four types in ADM: data base commands, scroll commands, workspace commands and edit commands. The data base commands are completely SEQUEL statements whereas the workspace and scroll commands provide extensive graphics capability in the system.

Advantages:

- Its display and image manipulation capabilities are quite powerful.

Drawbacks:

- The system is not machine-independent.
- The clustered organization of aggregate data in ADM causes a deterioration in its speed of operation.
- Since ADM is not designed with cartographic applications in mind, its capabilities for complex spatial operations are limited.
- The system is not designed to accept image-based information from a variety of sources.

Summary:

ADM is a modification or extension of a general purpose DBMS, SYSTEM R. Therefore, it does not have the power of a specially designed systems for specific applications. The main advantage in such systems is the relative ease of construction, as compared to special purpose systems. In summary, while the system has many useful capabilities as an image-oriented data base system, it is quite restricted in

its geographic capabilities.

3.2.6 GADS: Geo-data Analysis and Display System

GADS [17] is a decision support system that provides support for geographically related data. It was developed by IBM, San Jose, USA and is complete in its implementation. It is based on the relational data model.

In GADS, the geographic data is available in the form of an *event* data base and an *extracted* data base. Geographical interpretation of the event data is accomplished by reference to a corresponding polygon map file. GADS supports retrieval and display of data extracted from the event file, development and display of the extracted file from the event file, and aggregate operations on the polygon map. The event file is the source file containing relevant information while the extracted file basically contains data extracted from multiple event files.

The system also has two forms of geographic base files that contain data to support the relating of data from other files to geographic locations and also to display this data on a map. One is a table containing (x,y) coordinates for each of a list of geographic names/attributes. The other is a DIME file from which the (x,y) coordinates of any location can be developed.

The data base structure and management is based on the relational model. All the files are basically relations in the data base. The query capabilities are in the form of

functions or routines. The user can interact with the display through a light pen or a joystick-driven cross-hair, and an alphanumeric keyboard. The input data to the system, on the other hand, has to be in the defined format.

Advantages:

- It has adequate capabilities for feature representation in the form of event files, geographic base files and polygon maps.
- The system has useful interactive manipulation capabilities.

Drawbacks:

- The system is not portable, and can be used only on IBM machines.
- The full GADS system requires about 250K of memory, excluding virtual memory.
- It does not provide any facility to perform spatial operations on spatial data.
- The input does not accomodate information from a variety of non-imagery sources.

Summary:

In summary, the concept of "extraction" and the support provided for the extraction function are prominent features of GADS. However, the format requirements of the input data and the lack of image data display support are major drawbacks of the system.

3.2.7 GDB: A Geographical Data Base

GDB [13] is an interactive system designed to handle geographic information. It was developed by IBM, Mexico to run on their machines. The logical structure of the data base is relational. The query language is based on APL and consists of a set of APL functions. The data base management system is written in APL. The system has found use in many applications areas like agriculture, hydrology and meteorology. The system is complete in its implementation.

The information is classified into two types: positional and non-positional. The former refers to locational information about different points in a cartesian plane, while the latter refers to statistical information associated with a geographical position. The data base is viewed as a sequence of planes, each one of them containing one type of information. Therefore, a geographical point has a vector of attributes associated with it. This structure allows the use of views that are defined over any arbitrary subregion and through any combination of planes.

Data is classified in three types: zonal(region), linear(line) and punctual(point). Input data can be obtained from a variety of sources including:

(a) Landsat images,

(b) aerial photographs,

(c) cartographic maps and

(d) statistical tables.

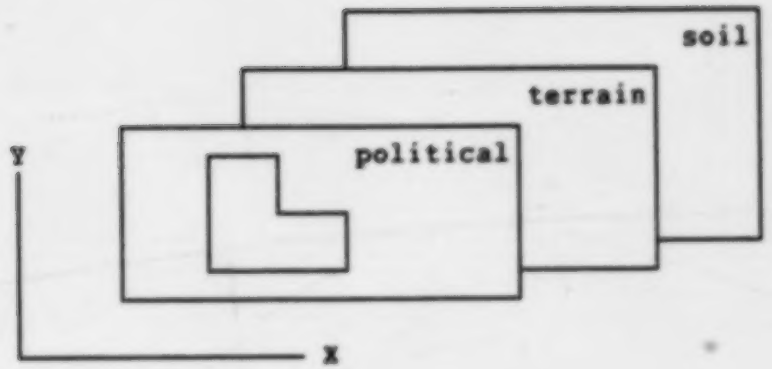


Fig. 3.2. PLANES OF INFORMATION

Two approaches are used to combine and integrate these different types of data: image-based approach which consists of representing information by means of matrices whose elements correspond both in position as well as in value to the different zones of interest, and the contour-based approach in which data is represented by means of coordinates of the vertices of polygonal lines approximating curves in the case of zonal and linear information and by coordinates of points of interest in the case of punctual information. The latter approach is advantageous in environments where memory limitations exist.

The output in the system can be either through a graphic display device or a line printer. The query language is based on APL and consists of a set of APL functions which is expandable.

Advantages:

- GDB can handle input from a variety of sources, and supports a range of output devices.
- It handles both spatial and aspatial data.
- It supports image processors to convert image-based features to a map-coordinate system.

Drawbacks:

- It is not clear how strong its capabilities are for incrementally updating feature descriptions based on updates to the image.
- GDB is not portable. However, with basic modifications it can be implemented on different machines.
- It is also not clear how inconsistencies are handled by the system, and whether the system can operate in a multi-user environment.
- The methods used in the synthesis of positional and non-positional components of data are also not known.

Summary:

GDB is an interactive geographic information system, based on the relational model of data base. One of the advantages of the system is that it supports input from a variety of sources and has image-to-map conversion features. It has some weaknesses too, one of them being that it is a specially designed system that is complex and costly to implement, as compared to a general purpose system. However, in spite of its limitations, the system has many useful capabilities, some of which are not found in many other

systems.

3.2.8 GEO-QUEL: A Special Purpose Geo-data System

GEO-QUEL [27] is a special purpose geographic information retrieval display system developed by the University of California at Berkeley. GEO-QUEL is very different from most other geographic information systems in that it is constructed on top of a powerful relational DBMS, INGRES. INGRES runs under the UNIX operating system and its source code is written in C.

In GEO-QUEL, all geographic data is treated as relations in INGRES. Therefore, most of the manipulation is done using the data sublanguage in INGRES and very little special code is required.

A basic entity in GEO-QUEL is a map, which is a collection of points, lines, line groups and zones. A map is stored as a relation in the data base. Besides a number of map relations, the system maintains a special relation-MAPRELATION, which contains the scale and center of each map for display purposes. Maps can be manipulated either by using the INGRES query language, QUEL, or by using special GEO-QUEL commands. A device-dependent routine, DISPLAY, produces a display list from the relations for display.

Advantages:

- The front end approach is easy to implement and requires considerably less code.
- It is flexible in that new capabilities can be added on

readily.

- It has considerably reduced software development cost as compared to a special purpose system.

Drawbacks:

- Because of its front end approach, the system has to suffer a loss in performance as compared to that offered by special purpose systems.
- The display routines in the system are device-dependent and changing the display devices necessitates rewriting these routines.
- The system has very few commands to perform spatial operations.
- Again, the system does not support any automatic data input capability. Instead, the data has to be manually input into the system in the required format.
- Because of the design constraint to store a map as a single relation, a certain amount of redundancy is present in the map relations.

Summary:

GEO-QUEL is a special purpose geographic information system that is constructed as a front end to a relational DBMS, INGRES. Therefore, the implementation approach of GEO-QUEL is very different from that of most other systems. One of the main drawbacks of the system is the deterioration in performance as a result of the implementation approach. This is compensated, however, by a dramatic decrease in the software development costs incurred.

3.2.9 MAPS: Map Assisted Photo Interpretation System

MAPS [18] is a large integrated data base system containing high resolution aerial photographs, digitized maps and other cartographic products and is used in many application areas like 3D scene generation, map-guided image segmentation, and image interpretation. It is being developed by the Carnegie-Mellon University. Although it is complete in its implementation, development work is still being carried out to incorporate more advanced features in the system.

MAPS is designed using an *image* or *map* data base model in which map features are related to an image data base through camera models. Therefore features acquired from different images can be related to one another in the system. It is not tied to a particular cartographic representation nor to limitations of cartographic production. MAPS is made up of approximately six major components, each of which further consists of several program modules. Important among them are BROWSE, the interactive image/map display system; CORRES, an interactive image-to-map correspondence program; LANDMARK, an interactive tool to generate new landmarks (ground control points); SEGMENT, an interactive image segmentation system; CONCEPTMAP, the data base component of MAPS; and MACHINESEG, a map-guided machine segmentation system. The BROWSE system provides a common interface to all the MAPS components to display the results graphically. The CONCEPTMAP data base

files contain the basic geodetic information about the maps. The Landmark data base which contains information about geodetic ground control points is part of an image-to-map correspondence process used to place new imagery into correspondence with the map data base. The other parts are LANDMARK and CORRES. Landmark features are also integrated into the CONCEPTMAP data base. CONCEPTMAP supports a number of spatial query operations on the data. The spatial data in the MAPS system is hierarchically organized. This helps partition the map data base to improve performance by decreasing search for spatial operations on map features.

Advantages:

- MAPS supports image-to-map conversion and correspondence.
- It also allows modifications and updates to the feature data.
- Complex spatial relationships can be represented in the system.
- MAPS has a very powerful map display capability, BROWSE.
- MAPS supports many synthesis tasks, including 3D scene generation.
- MAPS supports a wide range of query commands that are used to perform spatial operations on the data base.

Drawbacks:

- MAPS is quite complex in its structure, consisting of many subsystems.
- The creation of data bases in MAPS is also a complicated

task, involving more than one subsystem. In that sense the system is not user-friendly.

- The hierarchical organization of the data base, coupled with the large number of subsystems makes the memory requirements in the system higher than in many others.
- The hierarchical organization of the data introduces a performance penalty for search operations, in the system.
- Another limitation of MAPS specified in [18] by its designers is the method used to perform image-to-map correspondence. Its accuracy is dependent on the angle of view of the image and on the resolution of the image.

Summary:

MAPS is being expanded to include more complex features and make it a component of an image interpretation system. MAPS is a powerful geographic information system with many advanced features, but because of its size and complexity it is not sure as to how easily it can be used and adapted to different environments.

3.2.10 ARC/INFO: Computer Mapping and Geographic Information System

ARC/INFO [35] is a geographic software system that combines a geographic analysis and modeling capability with an interactive system for entry, management and display of spatial data. It was developed by the Environmental Systems Research Institute, USA and is fully operational.

The features in ARC/INFO include a command language, digitizing and entry subsystems, automatic editing and automatic creation of data bases, cartographic analysis tools like fast polygon overlay, point-in-polygon calculation, buffer generation, projection/transformation, network analysis, line and area measurement, and interactive table driven graphics. The commands are in natural language with names resembling the generic function desired by the user. Commands can be grouped to generate customized and application oriented procedures. The data base is organized using a relational and topological model. Data is classified into two types: cartographic data which describe the location and topology of features, and attribute data which contain the characteristics of these features. Attribute data is organized as relational tables in the data base. Cartographic data are structured with coordinates and topology. The latter is used to identify the relationships among arcs, nodes, polygons. The creation of a cartographic data base is fully automatic.

Advantages:

- ARC/INFO accepts input from a variety of sources like maps, satellite photos, non-topological graphic data from scanners etc.
- The system has a spatially indexed data structure for query and management of very large geographic data bases.
- ARC/INFO supports computer mapping and display

capabilities. Displays can be produced on b/w and color monitors and most models of pen and electrostatic plotters.

- It has a powerful query language. The query commands are in natural language and hence very user friendly.

Drawbacks:

- It is not clear whether the system is machine-independent.
- It is also not clear whether it can represent complex spatial relationships in the data base and whether inconsistency is recognized and handled.
- It has not been specified in [35] whether the system can relate feature acquired from different sources/images through the data base.

Summary:

ARC/INFO is a powerful, general purpose geographic information system. Since it has been specially designed, the development and implementation costs are very high as compared to front end or add-on systems like GEO-QUEL. The advanced features supported by the system compensates for this, however. Therefore, the system is best used in applications where a majority of its powerful features is necessary. In other applications of a limited nature, ARC/INFO would be uneconomical.

3.2.11 IDMS: Integrated Data Management System

IDMS [36] is a data base management system capable of handling both picture data and alphanumeric data. It cannot be classified as a specialized geographic information system in that it can handle any kind of picture and is not restricted to cartographic data (e.g., maps) alone. It was developed by the State University of New York and is complete in its implementation. The language SEQUEL is extended to serve as an interface between users and the system, in IDMS. The system is based on the relational data model.

In IDMS, the integrated data base consists of both alphanumeric data and picture data. The relations in the data base have been extended so that each attribute can have, apart from numbers and character strings, "pictures" or "devices" as a data type. The picture data type is represented by three numbers m,n,b such that m x n defines the size of a matrix and b defines the number of bits required to store each element of the matrix. Each picture attribute is syntactically independent of other picture attributes. This means that the occurrence of a picture attribute which is a subpicture of the an occurence of another picture attribute, is independent of the latter. Therefore, a subpicture can be accessed without getting its parent picture first. This, however, does not imply the need for physical data duplication. IDMS also provides a logical view of the I/O devices.

The I/O systems are represented in the form of relations in the data base. An attribute with the data type "device" specifies a device. The operating system then determines the physical device specified by each relation and acts accordingly. The data language used in the system is a modified version of SEQUEL. The features supported by the data language include query facilities, data insertion, deletion and update facilities, and data definition facilities.

Advantages:

- IDMS is a general purpose system and can handle different types of picture data.
- The system is simple and easy to use, since logical view of all the data is as relations in the data base. That is, alphanumeric data, picture data items, I/O devices are all treated at the level of attributes of relations in the data base.
- The system is designed such that changes in storage media and coding methods are invisible to the user.
- Data compression, which is essential for picture data, is accomplished by the data definition facilities in IDMS.

Drawbacks:

- IDMS is a general purpose pictorial DBMS that can be used in any kind of pictorial application, and hence does not provide some of the capabilities required for cartographic applications. It has limited capabilities to perform spatial operations such as containment,

adjacency and intersection.

- It does not have facility to convert image-based features to a map coordinate system.
- In IDMS, input data cannot be accepted directly from a variety of data sources.
- The display devices supported by the system and their display capabilities are not known since IDMS provides only a logical view of the I/O devices. The physical devices supported depend on the operating environment of the system.

Summary:

To summarize, IDMS is a data management system for an integrated data base consisting of picture data and alphanumeric data. The picture data is not restricted to only cartographic data. Because of this, many of the specific requirements of cartographic applications have been overlooked in order to achieve a general purpose system that can handle any type of picture data. Therefore, this system cannot be said to be ideal for cartographic applications.

3.3 Conclusions.

Over the last decade or so, as the volume of data increased steadily and the geographic information systems became more complex, there has been a perceived need to increasingly apply data base concepts in the design of these systems. However, "non data base" systems continue to be developed, especially when the systems are designed for

specific applications. Two such systems have been described in this chapter. Most of the systems based on data base concepts are themselves either special purpose [LUIS] or specially designed from scratch [MAPS]. Systems of the former type have limited usefulness, and while the latter type of systems are effective in their given environments, their design costs are often astronomical. This makes them uneconomical in applications that make limited use of the system. However recently, designers have begun to realize the economical advantages of constructing geographic information systems on top of existing general DBMSs [Geo-Quel,AQL, ADM]. Such systems do not usually support all the features of special purpose systems, but they combine the advantages of reduced implementation costs with the advantages of having the powerful capabilities of a well tested, general DBMS. One such DBMS which offers such extension (add-on) facility is described in the next chapter.

SUMMARY OF THE SYSTEM FEATURES

Feature	ALIS	ADAPT	LUIS	GADS	GDB	MAPS	ARC-INFO	IDMS	GEOQUEL	AQL	ADM
Data base concept	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Language supported	Fortran	Fortran	Basic	-	APL	-	-	-	C	APL	PL-1
Data model supported	-	-	Relat-ional	Relat-ional	Relat-ional	Image	Relational	Relat-ional	Relat-ional	Relat-ional	Relat-ional
Implementation	complete	complete	complete	complete	comp-lete	complete	complete	incomp-lete	complete	complete	comp-lete
Portability	yes	no	no	no	no	yes	yes	no	no	no	no
Query Language	Fortran routines	none	inter-active	routines	APL fns.	inter-active	interactive	SEQUEL	QUEL	based on APL	SEQUEL
Display capability	yes	mainly plots	yes	yes	yes	yes	yes	yes	yes	no	yes
Input sources	many	special format	many	defined format	many	many	many	many	-	none	only one
Developed in	Japan	USA	Canada	USA	Mexico	USA	USA	USA	USA	USA	Japan
System type	special purpose	special purpose	special purpose	general purpose	general purpose	general purpose	general purpose	general purpose	general purpose	general purpose	general purpose
Application	urban research	planning	planning	general	general	general	general	picture data	general	general	general
Operation mode	inter-active	batch	inter-active	inter-active	inter-active	inter-active	inter-active	inter-active	inter-active	inter-active	inter-active
Special hardware	3200K memory	-	-	IBM family	IBM family	-	-	-	-	IBM family	IBM family

CHAPTER 4

THE INGRES DBMS FOR GEOGRAPHIC DATA MANAGEMENT

In this chapter, the capabilities and drawbacks of the INGRES DBMS for geographic data is discussed. INGRES(Interactive Graphics and Retrieval System) is a relational data base management system which runs under the UNIX operating system [34]. It is implemented in C and the parsing of queries is done with the assistance of YACC, a compiler-compiler available on UNIX. It was designed at the University of California at Berkeley. The relations created in INGRES are stored as UNIX files.

The data sub-language supported by INGRES is QUEL (QUEry Language). QUEL is based on a tuple relational calculus [29]. Arithmetic is provided in QUEL, unlike some other languages which depend on the host language to provide this capability.

Although QUEL meets most of the data management requirements, a new language, EQUEL which consists of QUEL embedded in C is provided. This offers the flexibility of a general programming language in addition to the data base capabilities of QUEL.

INGRES supports five access methods [10] and these are based on three internal structures: *heap*, *isam*, and *hash*. Four of the schemes are keyed, i.e., the storage location of

a tuple depends on the value of the domains which are defined as the keys. The only non-keyed structure is "heap" in which the tuples are stored in a randomly ordered sequential file.

4.1 The INGRES Process Structure.

INGRES can be invoked either directly from UNIX or through the EQUEL precompiler [34]. In the UNIX environment, the INGRES data base system appears as a user job to the operating system. Consequently, the memory management is done by UNIX when INGRES is run as a job. UNIX also provides a facility for processes to share procedure segments, if possible. This facility is used by INGRES to minimize core memory overhead when there are multiple concurrent users on the system.

When INGRES is invoked directly from UNIX, the process structure shown in Fig. 4.1 is created [34].

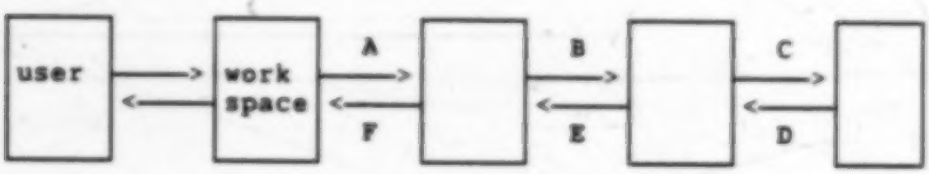


Fig. 4.1. The Ingres Process Structure

Process 1 is an interactive terminal monitor which serves as a user workspace to formulate and edit commands.

The contents of this workspace are passed through pipe A to Process 2. This process contains a lexical analyser, a parser, query modification routines for integrity and concurrency control. This process passes a string of tokens to Process 3 through pipe B. Process 3 contains execution routines for the QUEL commands like: RETRIEVE, REPLACE, APPEND. This process decomposes queries involving more than one variable into sequences of one-variable queries. One-variable queries are processed using a one-variable query processor (OVQP). All code to support utility commands (CREATE, MODIFY, INDEX, etc.) resides in Process 4. Process 3 passes to Process 4 any commands which the latter will execute. Error messages are returned through pipes D,E and F to Process 1 which then returns them to the user.

INGRES can also be invoked through the EQUEL precompiler. The process structure in this case is similar to the one shown in Fig. 4.1, except that the user module and Process 1 are replaced by an executable C program in which the QUEL statements are converted to appropriate C code.

4.2 The INGRES File Structure.

INGRES exists as a subtree of the UNIX file system [34]. Its root is the UNIX user **INGRES**. All the data bases exist under the directory **DATADIR**. The data base files are of four types:

1. administration file, which contains the user-id of the

DBA and initialization information;

2. catalog or system relations, which contain information about the relations in a particular data base;
3. DBA relations, owned by the DBA but accessible by other users unless protected;
4. relations created by other users.

The DBA has powers that are ordinarily not available to other users, including the creation of shared relations; specification of access control for shared relations, and the ability to destroy any relations in the data base.

Each relation is stored as a relation and no attempt is made to cluster tuples from different relations. Moreover every file under the INGRES directory is treated as a relation.

4.3 INGRES for Geographic Applications.

The INGRES data base management system is based on the relational model of data. The advantages and drawbacks of a relational approach to data base management have been analysed in Chapter 2. In this section, specific features of INGRES are considered for their usefulness in geographic applications.

In INGRES, new capabilities can be readily added. This gives the users the flexibility to construct any special purpose front-end to INGRES. This is made possible by the existence of a macro facility in INGRES, and by utilizing the EQUEL precompiler that embeds QUEL in C. In geographic

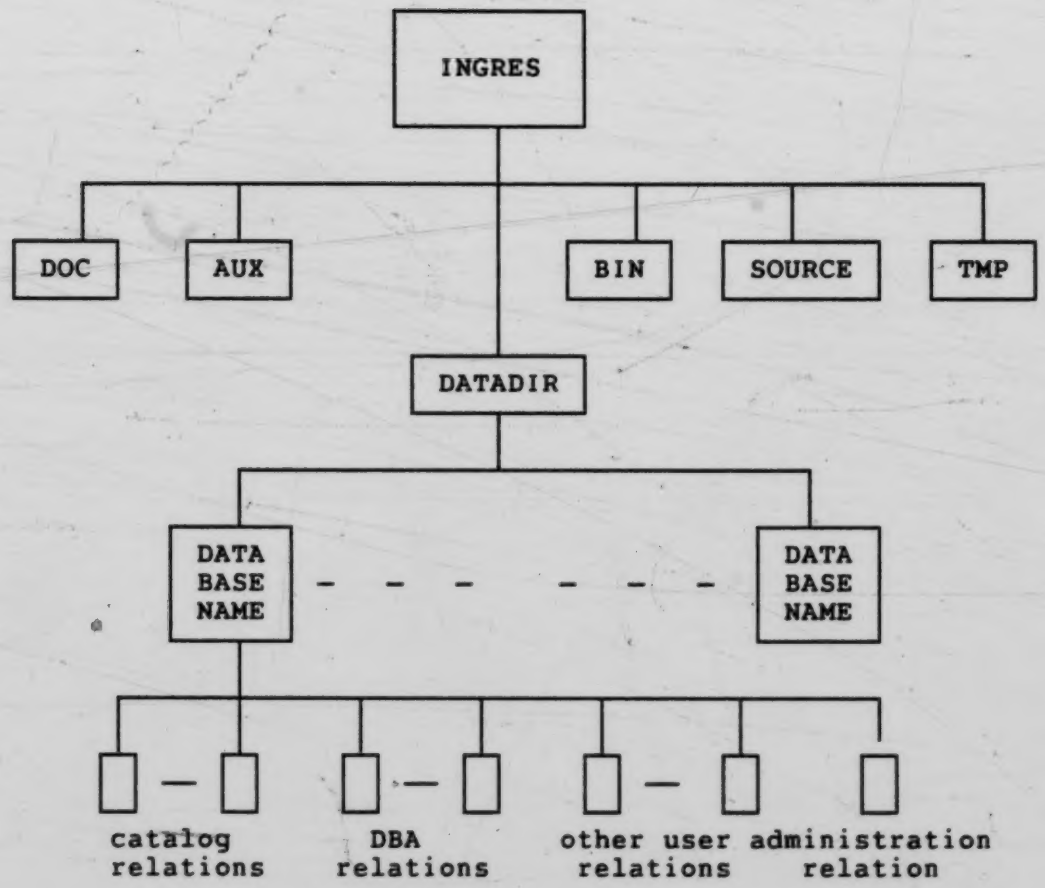


Fig. 4.2. The Ingres File Structure

applications, because of the diverse requirements of individual applications, it is not practical to either design or implement a generalized geographic data base management system that meets the requirements of all application areas. General purpose data base management systems like INGRES, with their flexibility to easily construct front-ends is a viable alternative in such situations. This property was used to implement GEO-QUEL, a

special purpose geo data system. The system implemented in this thesis uses the same principle.

The speed of operation in an INGRES environment is slower than that in many special purpose systems. This is because of the sequential organization of the INGRES process structure and also due to the number of translations of the code required in INGRES [see Section 4.1] in contrast to a special purpose system written in a lower level language.

This drawback is, however, offset to an extent by the INGRES facility that enables the users to choose any of the five storage structures supported by INGRES. These storage structures are designed to maximize performance under different conditions and users can select those that best suit their needs. It must be emphasized, however, that response time is a very important consideration in geographic information systems that support display capabilities. Interactive query and display facilities require fast response to user interactions.

The construction of a front-end to INGRES is fairly straightforward and standard in the sense that it involves the implementation of EQUEL routines that follow the same set of basic design guidelines. Moreover, this fairly well standardizes the design process.

INGRES does not provide any special data manipulation functions to support spatial data. Spatial and graphical data are treated as relations in the data base. QUEL, the data sublanguage supported by INGRES, has to be utilized to

manipulate the data.

A geographic data base system must be able to perform complex spatial operations such as containment and adjacency. Relational data base operations based on relational tuple calculus [29] supported by INGRES and its query language QUEL are inadequate for such complex operations. This is a common drawback of all relational data base systems.

Geographic data is basically of two types: *spatial* and *aspatial*. While INGRES, like most other relational systems, is very suitable for the storage and manipulation of aspatial data, its logical structure is inappropriate for the manipulation of spatial data represented in a cellular format.

Many of the spatial information systems are designed to have a hierarchical organization, including the one presented in this thesis. This is done in order to improve performance by decreasing the search time and satisfy some data constraints.

Fig. 4.3 is an example of the hierarchical organization of geographic information.

It must be remembered, however, that the logical structure of the relational model is not an ideal primitive for the implementation of such hierarchically organized designs. Therefore, INGRES has many of the disadvantages that are associated with the relational data model on which it is based. On the other hand, it has some useful features

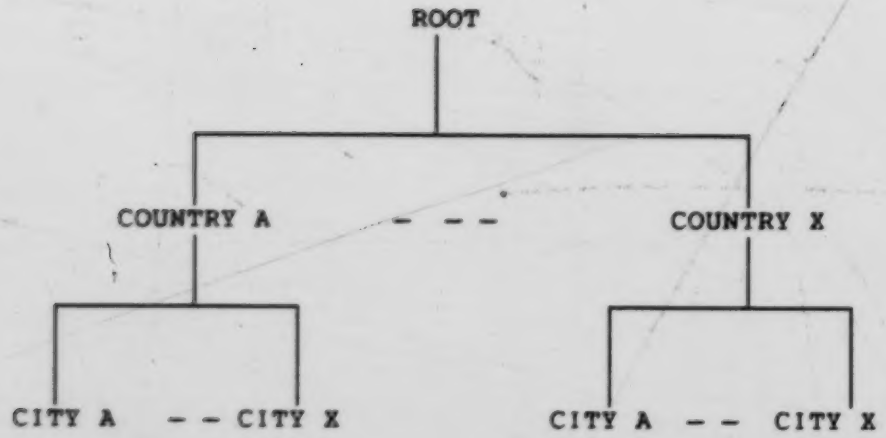


Fig. 4.3. Hierarchical structure of spatial information

as far as geographic applications are concerned, including the front-end add-on capability, the powerful capabilities of QUEL and the ease of implementation.

4.4 Summary

In this chapter, the capabilities of the INGRES DBMS for geographic applications has been discussed. Details regarding the design and implementation of INGRES have also been presented. The next chapter deals with the implementation of a specific geographic information retrieval system on INGRES.

CHAPTER 5

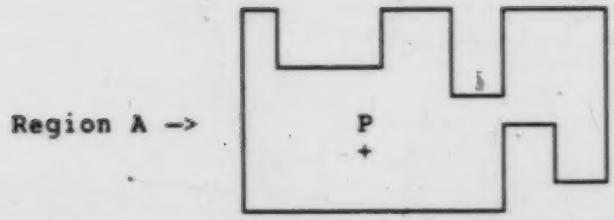
DESIGN AND IMPLEMENTATION OF A GEOGRAPHIC INFORMATION RETRIEVAL SYSTEM

In this chapter, the design and implementation of a geographic information retrieval system are delineated. This system is constructed as a front-end to the relational data base management system, INGRES, running under the UNIX operating system.

5.1 Scope of the System.

Like most other geographic information systems that are based on the relational model of data, this system is restricted in the geometric operations that it can perform. This is further restricted by the fact that this system is constructed using a front-end approach (the disadvantages of the front-end approach are explained later, in this chapter). Some specific observations regarding the geometric capabilities of this system are enumerated below:

1. Point-Point operations can be performed without any restriction.
   e.g., What is the distance between Point X and Point Y?
2. Point-Region operations are restricted to regions that are regular shaped geometric figures like rectangle, square etc. For example, it cannot handle a query like,



Is Point P contained by Region A?

3. Line-Region operations are also restricted to regular shaped regions.
4. Region-Region operations can be performed only if the participating regions are regular shaped geometric figures.

The above constraints essentially define the scope of the Data Base module, and should not necessarily be construed as drawbacks of the system.

5.2 System Architecture.

In this section, details regarding the architecture of the system, the design of the various components and the implementation of the system are discussed in detail.

5.2.1 Introduction.

The system is composed of three basic modules - the File System Kernel module, the UNIX File System module, the Data Base module. Together, these three modules have been termed the "data management subsystem". This subsystem is,

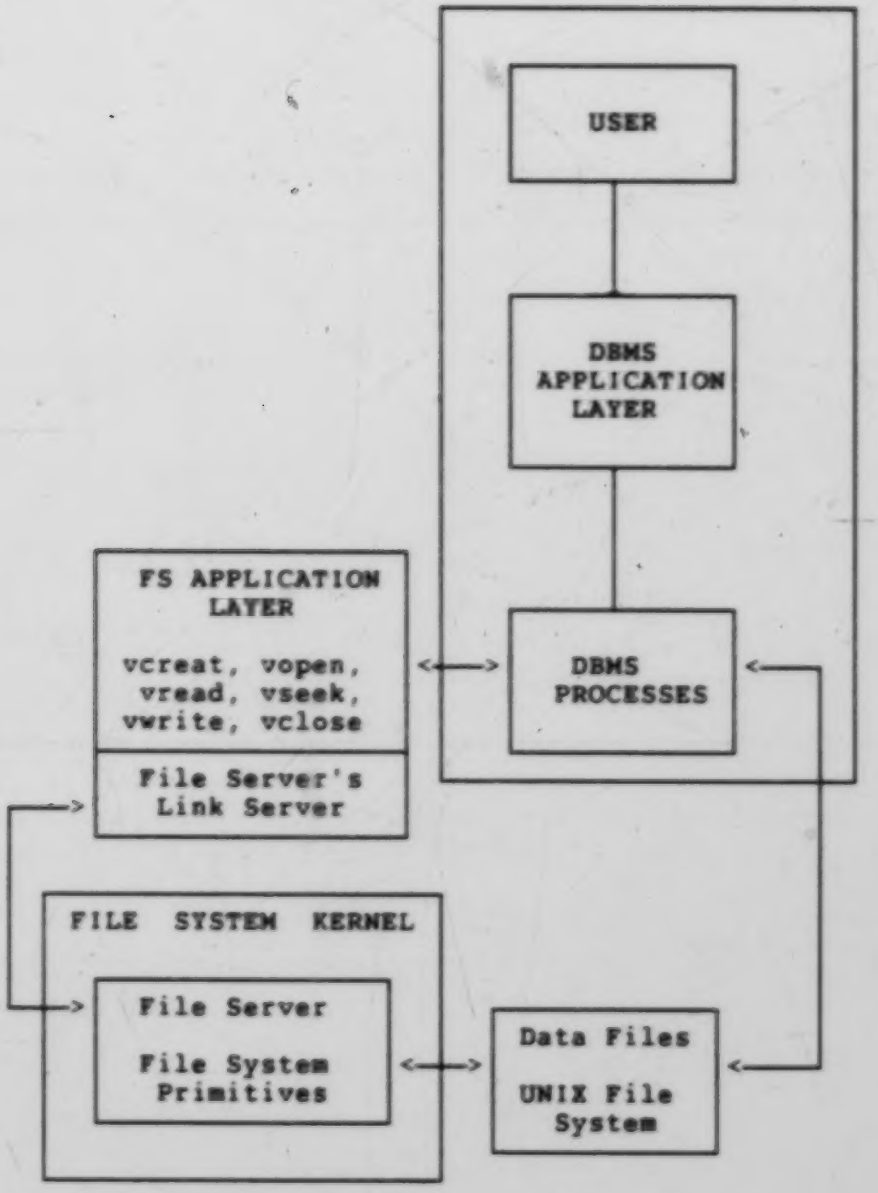


Fig. 5.1. The overall architecture of the Data Base and the File System Kernel

in turn, interfaced with two other modules - the Graphics module and the Communicaions module, to form a complete system that can be used to retrieve, display and manipulate geographically distributed pictorial data.

The Communication module embeds itself in several layers of the system, from the busses to the remote data links. This module is concerned with the establishment and management of communication links between display to host computer, host computer to secondary host computer, and host computer to remote data sites. The Graphics module is concerned with the graphical display of retrieved pictorial data. This module also includes image processing operations such as image smoothing, edge detection, boundary following and other enhancements. The File System Kernel module provides a set of functions for accessing data within the file system. In short, this module serves as a "low-level" data management subsystem. Providing a "higher-level" data management service to the user is the Data Base module, through which the user interacts with the subsystem. The Data Base module, in turn, interacts with the File System module to satisfy the user requests.

In the next few sections, various components of the "data management subsystem" are explained in detail.

5.2.2 File System Kernel Module.

The File System Kernel [see Fig. 5.1] is a set of functions for accessing data within the file system [38].

Via file system primitive calls, the user initiates requests to the file server through the DBMS Application Layer (front-end), which is a part of the Data Base module. The file server is accessed by the file server link server. The file server utilizes the file system's primitive functions in order to process the caller's request and access the necessary data file(s) within the UNIX file system. The result of the access is then passed back along the reverse path to the Data Base module.

The Data Base module calls the File System Kernel when data required to process a specified query is stored as data files in the file system. When the required data is stored as data relations (to be explained later) in INGRES, the file system is not called. Instead, the data relations, which are physically stored as files under UNIX, are directly accessed by INGRES. A 1 bit flag (0 or 1) in the index relation (to be explained later) indicates whether the required data is stored as a file or as a relation.

To summarize, the file system kernel is called by the Data Base module when data required to process a query is stored as files in the UNIX file system.

5.2.3 UNIX File System Module.

UNIX provides a hierarchical file system, consisting of three types of files - ordinary disk files, directories and special files. Ordinary files contain whatever information the user places on it. They are created, manipulated and

destroyed by the user. Directories provide the mapping between file names and the files themselves and thus provide the hierarchical structure of the system, as a whole. Special files refer to different I/O devices since, in the UNIX file system, I/O devices are treated as files. There are several literature on the UNIX file system, including [40].

The actual data is stored under the UNIX file system. All the relations in INGRES are physically stored as files under the UNIX file system. However, the user can only manipulate these files as relations. Data relations, on the other hand, can be explicitly stored and manipulated as UNIX files by the user.

In summary, the actual data is stored under the UNIX file system, and is accessed either by the File System Kernel or by the Data Base module, depending on how the data is stored.

5.2.4 Data Base Module.

The Data Base module provides an interface between the user and the file system. This module provides a set of functions for the user to manipulate the data that is stored under the file system. The Data Base module, in turn, activates the file system to satisfy the user requests.

The Data Base module consists of three sub-modules - the User module, the DBMS Application Layer, and the DBMS Processes module. The User module provides the user with a

set of functions (in the form of a query language) to manipulate the data. The DBMS Application Layer contains the actual implementation of these query language functions. Therefore, whenever the user issues a command to the User module, the corresponding functions in the DBMS Application Layer are called the by former. The DBMS Application Layer, in turn, invokes the DBMS Processes module, which contain the the three INGRES process modules (explained in Chapter 4). The DBMS Processes module interacts with the UNIX File System and the File System Kernel module of the system to satisfy user requests. The overall architecture of the system is detailed in Fig. 5.1.

5.2.4.1 System Requirements.

The functions of the Data Base module, as far as the user is concerned, can be broadly classified into the following:

1. *Searching:* Given specified data characteristics, find and return the requested information from the data base.
2. *Locating:* Given specific data characteristics, output the location of the specified entity.
3. *Updating:* Whenever a file (relation) is created, deleted, renamed or modified, update the corresponding entries in the index relations so that there are no inconsistencies in the data base.
4. *Checking:* Whenever a query is executed, check for possible errors both in the syntax and the execution

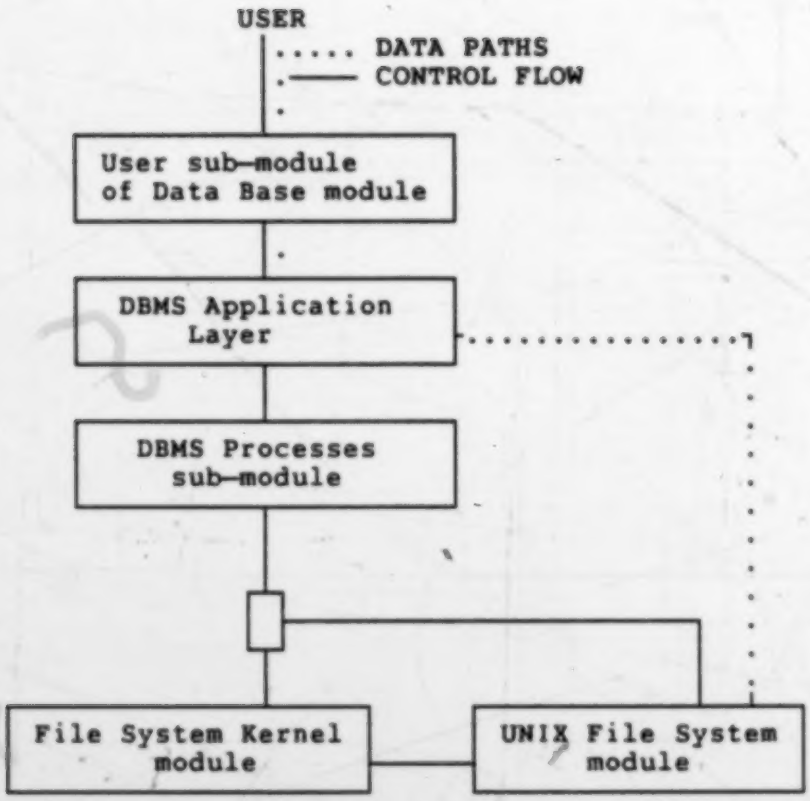


Fig. 5.2: Block structure of the execution of user commands

paths of the commands.

5.2.4.2 Proposed Hybrid Data Structure.

From the analysis of the three data models in Chapter 2, it has been determined that none of the them are ideally suited for the type of geographic data base under consideration. Based on the observations in the same analysis, a new relational/hierarchical hybrid data structure has been proposed and used in the design of

this data base.

As noted in Chapter 4, geographic information is, in general, hierarchical in nature and therefore a pure relational approach is not particularly suitable for their design. The other alternative, namely the hierarchical approach, is unsatisfactory for reasons elaborated in Chapter 2. A relational/hierarchical hybrid approach, on the other hand, combines the advantages of the pure relational approach with those of the pure hierarchical approach.

The basic structure of this hybrid approach is a hierarchy, in which each node may have one or more successors and exactly one predecessor. Each node in the hierarchy, in turn, consists of one or more relations in the data base. And, each relation in a node can be any of the three types that are defined for such a hybrid structure. The three types of relations that are defined are:

1. **data relations**, which contain the actual data either in the form of relations in the data base or as data files in the UNIX file system,
2. **index relations**, which contain information about other relations (more specifically, information about all their respective successor-relations at the next level in the hierarchy) and
3. **directory relations**, which maintain information about all the relations in the data base.

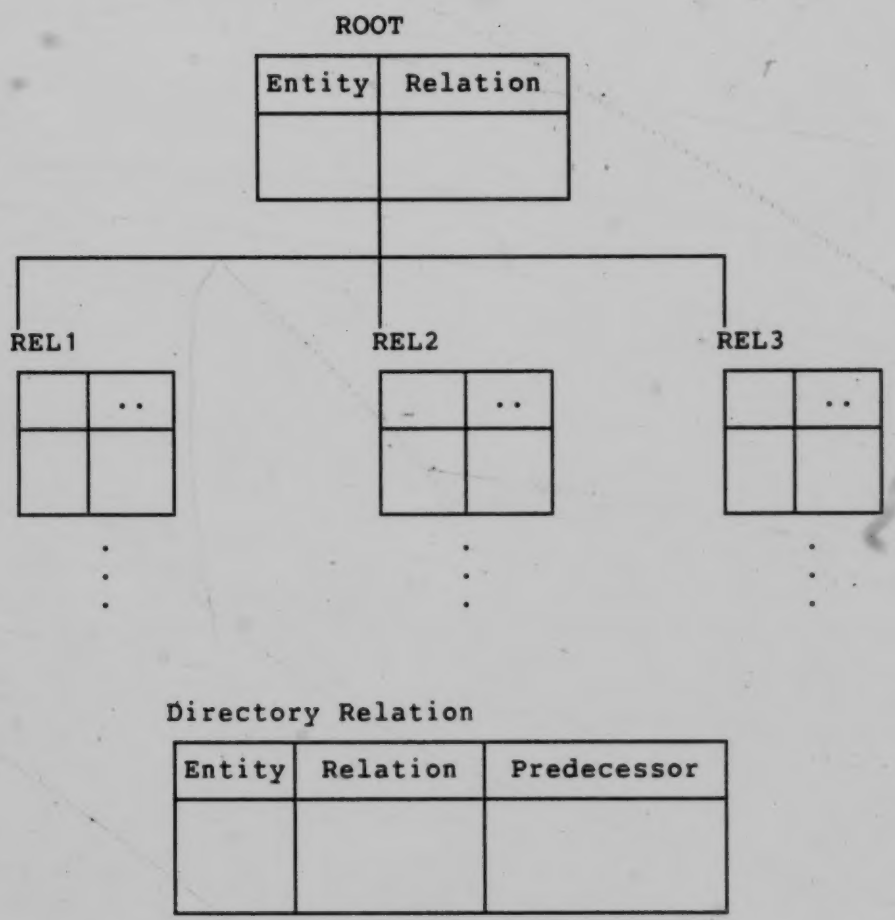


Fig. 5.3. Proposed Data Base Structure

This information includes the name of each geographic entity, the name of its corresponding relation in the data base and the name of its predecessor relation. The information about the predecessor relation is important in providing a hierarchical conceptual view of the data in a relational data base.

Because of the hierarchical structure, each node has exactly one predecessor. Thus by recursively identifying the predecessors, an entire path from any given node to the root (of the hierarchy) can be determined. This is necessary in search operations where any particular entity may be associated with more than one corresponding relation at different locations in the hierarchy, in the data base. In such instances, the determination of the location of a relation in the hierarchy is critical. For example, the entity name "London" may be associated with two cities - one in Ontario (Canada) and another in U.K. A hierarchical structure is very useful in distinguishing between relations or file with identical names corresponding to two such cities with the same name. This can be done by specifying a particular predecessor (like, country = "Canada").

5.2.4.3 Data Base Query Language.

The data base query language is implemented as a front-end to the INGRES DBMS. Each query command is an EQUEL routine, implemented in C and QUEL, that calls the INGRES process for execution. An entirely new set of query commands was required because the specialized nature of the data base (i.e., a geographic data base) cannot be supported by the set of commands in QUEL.

The set of commands includes:

1. Commands to locate specific relations in the data

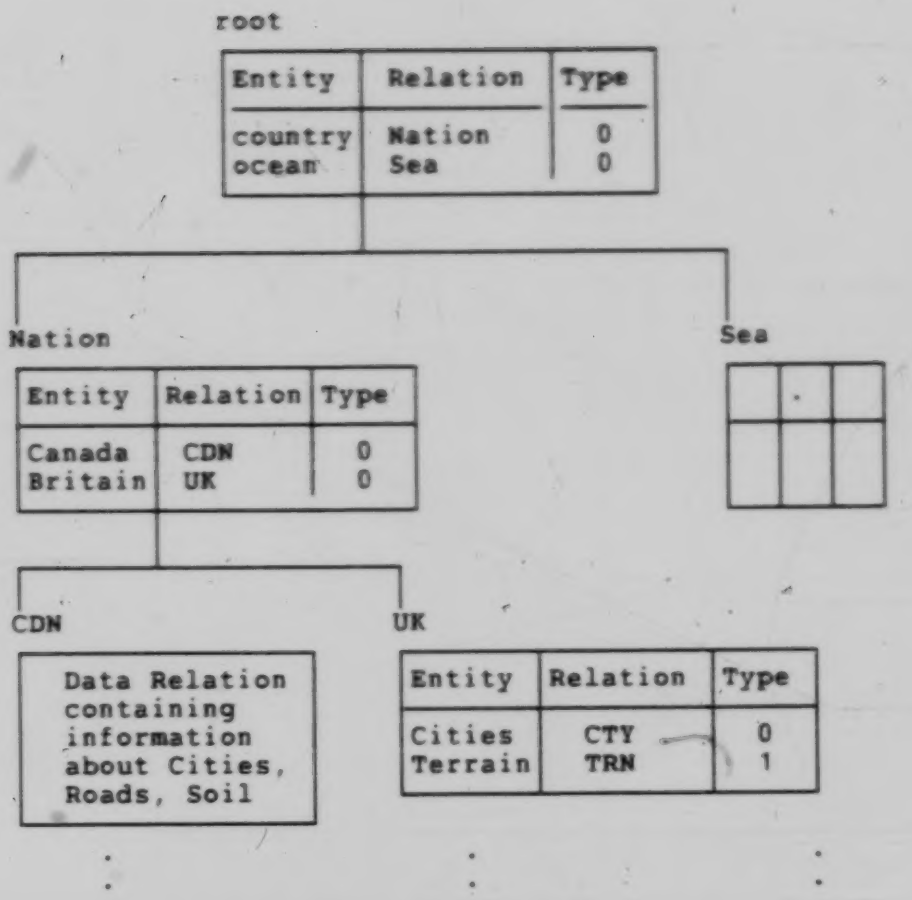
base.

2. Commands to retrieve relations corresponding to a specified geographic attribute (or entity).
3. Commands to print out detailed error messages in the system.
4. Commands to traverse and/or search the hierarchical structure of the data base.
5. Commands to search and retrieve information from the data base based on specified qualifications. The qualifications can be complex and may include the specification of the path in the hierarchy or the specific value that a predecessor should have.
6. Commands to create new relations at any level in the hierarchy. The created relation is linked to its predecessor.
7. Commands to delete records and destroy relations. Such commands are consistent in that they delete the corresponding entries in the predecessor relation(s) whenever any deletion is made.

Example 5.1 is a sample data base using the hybrid organization. Consider the following query on this sample data base:

Find name of Cities for country = Britain where population LT 40000

The system would first retrieve relation names corresponding to 'Cities', 'country' and 'Britain'. It would then generate the different paths from any relation whose entity name is



Directory Relation

Entity	Relation	Predecessor
country	Nation	root
ocean	Sea	root
Canada	CDN	Nation
Britain	UK	Nation
Cities	CTY	UK
Terrain	TRN	UK

Example 5.1: A sample data base query

'Cities' to the root. Next, the system selects that path where the qualifier "country = Britain" is satisfied. It then accesses the corresponding 'Cities' relation and

retrieves the required information. Note in Example 5.1 that both CDN and UK contain information about their corresponding cities, but in very different formats.

5.2.4.4 Query Language Considerations.

The main emphasis in the design of the system is on its ease of use. The amount of specification required from the user is designed to be a minimum. The assumption in the design is that the user has only a bare minimum amount of information that is required to perform the required operations. Thus, even a casual user should be able to operate the system with sufficient ease and comfort.

Again, the specification (information) required from the user is designed to be as simple as possible. As a result of these considerations, error messages are designed to be extensive and understandable.

For the experienced user, however, the full range of the query capabilities of INGRES is available to perform operations on the data base. This increases the capabilities of the system by almost two-fold and represents a very powerful facility in the system.

In fact, the query language set can be conceived as a two-layer interface. The lower layer consists of the INGRES query set and the upper layer is the query set designed specifically for this application [see Fig. 5.4]. Again, the speed of operation, and hence the response time, are important considerations in the

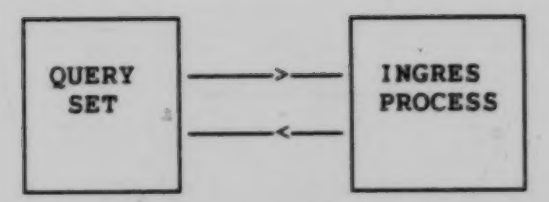


Fig. 5.4. Two-layer query language interface

design of the query language. The response time is significantly influenced by the physical organization of data in the data base. INGRES permits a variety of data organizations to improve performance in search and retrieval operations on the data. In the query language design, performance considerations include the minimization of calls to trigger (start up) the INGRES DBMS, minimization of retrieve operations in the data base, and efficient design of the routines such that any error will cause an immediate exit from the routine (with proper error messages) without any further processing. For example, consider the routines in Fig. 5.5.

```
Routine  1                    Routine  2
Ingres call                   Ingres call
........                      ........
exit                          ........
Ingres call                   ........
exit                          exit
```

Fig. 5.5. Two Ingres routines compared

Routine No.1 causes a deterioration/delay in performance by almost twice that of Routine No.2 because it calls INGRES twice as many times as does Routine No.2.

Another consideration in the design is *consistency*. That is, the language operators must be consistent throughout the command set. Operators should not change semantics in different contexts. The design of the language syntax is made to be simple, but covers a variety of complex transactions. In fact, the language syntax has been designed to be as close to the syntax of the English language as permissible under the design environment.

Even the command-line parameters reflect this design consideration. For example, the user does not have to know the name of any desired relation. Instead, the user can extract this from its English language equivalent. Suppose the user requires information about islands. The user does not have to know the name of the corresponding relation, say, "PLN.ISLAND" but only its English language equivalent, "island".

Most of the commands in the query language are single line commands. Another consideration in the language design is the sensitivity of the language to changes. That is, a small change in the query language expression must not require the rewriting of the query

with an entirely new structure. This is clearly reflected in the formulation of the search queries in the command set. Easy detection of errors is another useful design consideration. The query structure minimizes the possibility of errors. If, on the other hand, the user formulates an erroneous transaction, a simple and clear error message is given by the system.

All these factors have been important considerations in the design of the query language.

5.2.4.5 Advantages of a Front-End Approach

This geographic information retrieval system uses a front-end approach in its implementation. There are several advantages in such an approach and these weighed heavily in the decision to adopt it in this system. The advantages include:

1. The full power of the relational query language is available to manipulate the data.
2. Because of 1 , most operations on the geographic data can be converted into interactions in the query language and hence the amount of special code required is substantially less.
3. Considerably reduced system development time and cost due to 1 and 2.
4. There is considerable flexibility to add new capabilities and/or modify existing ones with relative ease. In this system, for example, any new set of commands can be added without modification to

the existing system.

5. The ability to utilize different software packages supported by the parent system is available. This represents a very significant advantage over a special-purpose approach. For example, a geographic information retrieval system constructed as a front-end to INGRES can use the full power of the programming language C, along with all the associated software packages supported by it. In a special-purpose system, each software package has to be developed either from scratch or by extensive modification to the existing ones.
6. Convenient access to special-purpose hardware, including image processing and graphical devices, supported by the parent system. To illustrate, a system constructed as a front-end to INGRES and running under UNIX, can call using C a wide range of image processing and graphical devices.
7. Concurrency, integrity and security requirements are better met in a front-end approach where the powerful, tested facilities of the parent system can be utilized.
8. It is easier to distribute systems built around a general purpose DBMS than a special-purpose one. The different mechanisms required for distribution are more easily available for the former type. Moreover, standardization considerations for the different

sites of a distributed network are easier to handle in a general-purpose system.

5.2.4.6 Disadvantages of a Front-End Approach

Although the front-end approach has several advantages, it has a few drawbacks that must be mentioned. The drawbacks include:

1. A system built using a front-end approach may not be able to perform certain operations as fast as a special purpose system using special purpose code in a lower level language.
2. The spatial operations that can be performed in a front-end system is limited by the query language capabilities of the parent system, on top of which the front-end is built. In many cases, the query language of the parent system is not designed for geographic data.
3. The degree of extension that can be done to an existing system to make it a front-end system, for geographic data, is limited by flexibility offered by the structure of the parent system. For instance, some existing DBMSs allow their query language to be easily modified to suit the requirements of specific applications while other systems do not provide such flexibility.

5.2.4.7 Performance Considerations.

1. Physical data organization: INGRES supports nine different storage structures. They include ISAM, CISAM, hash, chash, heap, cheap, heapsort, cheapsort and truncated. A relation is always created as a heap. However, suitable storage structures can be selected to augment performance. The selection of a suitable storage structure depends on the how the relations are used and is outlined in [10]. These guidelines have been followed in the selection of the storage structures in this system.
2. Calls to INGRES: Each time a call is made to INGRES, the system has to check whether the user is a valid INGRES user and whether the user is permitted to access the requested data base. This: takes up considerable time, as a result of which performance suffers. Therefore the number of calls to INGRES are kept to a minimum.
3. Repeated commands: INGRES provides a facility whereby repeated commands can be "pre-compiled". That is, the first time a command is used, it is compiled and the subsequent executions of the command are done using the compiled version. This improves the response time for queries.
4. Retrieve loop: In INGRES, the "Retrieve" loop should be used to retrieve only sets of records. The system performance suffers dramatically when this command

is used for single record retrievals.

5. Exit on errors: Another performance consideration is to cause an immediate exit when an error is detected in the execution of the query.

5.3 Conclusions.

This system has been implemented on a VAX 11/780, running under the UNIX operating system. The software was written in C and QUEL.

Details regarding the data base and the reference manual for the query language are given in the appendices.

CHAPTER 6

DISTRIBUTED GEOGRAPHIC DATA BASES

Spatial data bases are, in general, geographically distributed and the spatial data sites in Canada are an obvious example of this. These include: topographic and geographic data in the Surveys and Mapping Branch of Energy, Mines and Resources; aerial and satellite multispectral scanner data at the Canada Centre for Remote Sensing of EMR; forestry and meteorological data at Environment Canada; agriculture data at Agriculture Canada; statistical data at Statistics Canada; plus the corresponding sources in each of the provinces. However, geographic distribution is usually a necessary but not sufficient reason for considering a distributed system [5]. There are several other factors that have to be taken into consideration before a distributed system can be justified.

Before the reasons for considering a distributed system are enumerated, it is useful to identify the concept of distributed systems. A distributed system is, basically, a system with processing capability and software dispersed throughout a number of sites linked by real-time communications. The distinction, however, between a *distributed processing system* and a *distributed data base system* must be clearly understood [5]. In the former, the

data base is generally assumed to be centralized whereas in the latter, the assumption is that portions of the data base are dispersed throughout a number of sites linked by real-time communications. In a distributed geographic system, each site is associated with large volumes of data which makes it impractical to have a centralized data base configuration both from communications cost considerations and performance considerations. Therefore, the assumption, in a distributed geographic system, is reasonable that the data base is distributed. The distributed data base (DDB) provides a uniform view of the component data bases to all the users, irrespective of where they are located. The users see only a logical collection of data and they need not know the location of an object to utilize it.

Such a data base can be divided into the following components [5]:

- The schema, which describes the data structures and logical data base organization.
- The actual data.
- The application programs which access and utilize the data.

6.1 Motivation for distribution.

The motivation for distributing data over the sites, compared to the centralized or decentralized (stand-alone) approaches, involves many considerations : hardware, communications, management, software, performance etc. These

include the following [5,12]:

a. Performance tailoring.

Centralized systems must be tailored to meet the requirements of a variety of applications. However, in practice this is not always possible without degrading performance. For instance, in the example mentioned earlier, statistical data at Statistics Canada has totally different requirements than the forestry and meteorological data at Environment Canada. Hence, centralized systems are tuned to maximize performance for the most commonly occurring workload types. Because of this consideration, the less common and specific needs of individual users may not be met.

b. Format restrictions.

Large centralized systems usually make use of only one type of data format (vector or raster). The data structure and organization are also usually fixed, in a centralized configuration. This imposes restrictions on the use of the system and reduces the flexibility available to the users. In a distributed system however, each site can have its own structure and organization. This of course would require the use of conversion routines when called by remote users whose host site have different structures or formats.

c. Volume concentrations.

In a centralized system, large volumes of data are concentrated at one location. In a geographic information system, the volume of data is exceptionally high. In a distributed system however, the data is distributed over all sites. This lessens the need for complicated structures and consequently increases the performance, for local operations.

d. Specific requirements.

In a distributed network, each site can be geared to the specific needs of that location. This is an important consideration in geographic information systems where different sites are designed for different applications. Data base sites in Canada are an obvious example of this.

e. Communication costs.

In most situations, considerable economies can result by processing data at the point of capture, avoiding the need to transport data to a remote central location for processing.

f. Site failures.

The risks associated with system failure are generally considerably less in a distributed configuration than in a centralized system. In the latter, the failure of the

system can result in a total shut-down of processing activities.

g. Decrease in complexity.

Distributed data base systems designed in such a way that the majority of enquiries can be handled and processed locally can provide a relatively simple and elegant solution to the problem of data access, especially when compared with centralized data bases mounted on large multiprogramming and multitasking systems.

h. Faster response to queries.

In a system in which data is distributed over all the sites, data required to process a majority of the queries is generally available at the local site itself. This reduces the average response time for queries.

i. Addition of new spatial data base sites.

The centralized configuration has limitations(memory limitations, processing limitations etc.) on the number of users/sites that can be added to the system without upgrading the hardware. The distributed sites, on the other hand, are locally autonomous and the addition of new sites do not affect the existing ones to any great extent. This, in fact, improves the modularity of the system.

A distributed system clearly offers many potential advantages, and its underlying technological and organizational concepts are fundamentally sound. Nevertheless, a number of hazards exist in a distributed system.

Duplication of data and synchronization are potential problems in distributed data bases. In a geographic information system where the data at each site is basically the data captured at that site and where the frequency of update is low as opposed to, say, an airline reservation system, duplication and synchronization are not serious problems.

The distributed concept can add significantly to the security risks involved. This would probably be a serious hazard, for instance, in a banking system where protection against fraud and embezzlement is of primary concern. This is not the case in a geographic system where the data is used mainly to display information traditionally represented in the form of maps.

A key concern in a distributed system is the potentially unlimited design complexity. Distributed systems exist in an unending set of combinations of system architectures, communication topology and nodal configurations and hence, a rational system design and implementation can be very difficult.

Inspite of these potential problems, the benefits of distributing the system outweigh the drawbacks that result

from the same, at least as far as a geographic information system is concerned.

6.2 Distributed Data Base Architectures.

This section briefly describes the basic types of distributed data base architectures [5]. The first type, a centralized data base and distributed users, involves minimal distribution. The second type is the partitioned data base. The third type is the replicated data base in which the data base is replicated at a number of sites.

6.2.1 Centralized Data Base, Distributed Users

In this type of architecture, a single centralized copy of the data base is maintained. However, the users of this data base can be both local and remote. Since the data base is not distributed, most of the DBMS functions reside at the central site.

For retrieval operations, some initial processing may be done at the remote site before the request is passed to the central site where it is processed. It is also usually necessary to perform schema-to-user schema conversion either at the remote or the central site before a response can be obtained at a remote site. For update operations, the required data is sent by the remote site to the central site, which sends back an acknowledgement. This too requires a user schema-to-schema conversion. Because of the centralized nature of the data base, this type of

architecture is not considered to be very advantageous in distributed applications.

6.2.2 Partitioned Data Base

In a partitioned data base, the single copy of the data base is partitioned into non-overlapping parts and each part is located at a different site. Different criteria can be used to partition the data base, like data value, data item, etc. Each site contains a DBMS, a partition of the data base, and the schema describing that partition, among others. Each site however, does not necessarily contain all components. Some sites can be simple remote user sites with no processing capabilities. The system also needs some kind of data base directory for the entire network to identify the location of the different data base partitions. The operations of this type occur in three types: local, remote and mixed. In the local type of operation, all the required data is available at the local site itself. In the remote type, all the required data is located at a single remote site. In the mixed type, parts of the requested data is available locally and parts are available at one or more remote sites. The requests are handled by the DBMS which, in consultation with the data base directory, determines the location of the requested data. In the case of a mixed type of request, additional functionality is required to decompose the query, select the processing strategy, and aggregate or do any additional processing of the partial

responses. The mixed type of request also complicates the locking, backup and recovery requirements in the system.

In practice, this system can be of two types:

1. a single logical data base that is partitioned based on the activity against various parts of the data base at different sites, or
2. several independent data bases that are connected together.

6.2.3 Replicated Data Base

In this architecture, parts of the data base are replicated at a number of sites. This type is an improvement on the partitioned type for the case where there is insufficient locality of reference for the requests. Under such conditions, replication provides improved response time and increases reliability against failures. On the other hand, replication increases the storage requirements in the system. Furthermore, because of the replication of data, issues such as synchronization, concurrency of updates require efficient solutions.

6.3 Distributed DBMS Functions

A Distributed Data Base Management System (DDBMS) is a set of programs implemented at various sites of the distributed network and is responsible for supporting and controlling user interactions with the distributed data base. The functions of a DDBMS include most of those

provided by a conventional centralized DBMS. In addition to these, the DDBMS also provides some functions to support data bases in a distributed environment. These include the following [5]:

1. The DDBMS should identify the complete access path of any data reference in a query. That is, the DDBMS must be able to identify the logical site addresses of any data reference in order to send a request to the proper site in the network.
2. The DDBMS must provide the linkage between different components of a distributed system. All requests initially go to the DDBMS which then directs them either to the local DBMS or to the communication subsystem, depending on the location of the data.
3. The DDBMS is responsible for determining the strategy to process a query. The DDBMS must decide whether all the required data is available locally or whether data from some remote site is required to process the query. It must then route the query to the appropriate subsystem. For complex queries requiring data from more than one site, the DDBMS must decompose the query and send the different parts to the appropriate sites.
4. The DDBMS should provide recovery from site failures. If a site in the network fails, the DDBMS should maintain operations at other sites, as far as possible. Different situations can arise when a site fails during processing of a transaction. The DDBMS must be able to handle all

situations.

5. The DDBMS must provide consistency of data and also protect the integrity of the data base.
6. In a heterogeneous environment, the DDBMS must provide mapping or translation facilities between sites that are dissimilar.

So far, the distributed data base systems have been discussed in general, without much reference to any specific type of application. In the following sections, however, a specific type of application, geographic information retrieval, is considered and the issue of distributing data bases in such an environment is explored.

6.4 A Distributed Geographic Information Retrieval System

In this section, the extension of the geographic information retrieval system detailed in Chapter 5 to a distributed environment is discussed in detail. All the basic assumptions and features of the system, specified in Chapter 5, are valid for the distributed environment, unless stated otherwise. It must also be mentioned that the emphasis in the design is on the distributed data base system and not on the network. In other words, details regarding the network topology, communication protocols etc. are not of concern here. The assumption in the system is that the network topology is such that each one can communicate with all other sites in the network.

6.4.1 Data Distribution Strategy

The data distribution strategy refers to the generic architecture of the distributed network. The three major distribution strategies have been described in Section 6.2. A fourth, the hybrid, is a combination of *replicated* and *partitioned* strategies. The selection of the appropriate strategy depends mainly on the number of sites containing data and the degree of data duplication present in the network.

The distribution strategy adopted in this system is a hybrid one. It is hybrid in the sense that it is a combination of the partitioned and the completely replicated strategies. The hierarchically organized data base is split into logical fragments, each fragment consisting of one or more relations. A relation is not split into smaller parts and is therefore the smallest element of distribution. In geographic applications, each type of data is collected or assembled at a single site. It is not usual for more than one site to collect the same data. And, since in this data base, each type of data is represented by one or more relations, it was not considered necessary to split up a relation into smaller fragments. The enormous increase in the complexity of the system that would result from the division of relations into smaller fragments was another major factor in the decision against such a split.

Flexibility is one of the important advantages of this strategy. Trade-offs can be made between the various

requirements in a distributed system. For example, data can be duplicated as desired, depending on the locality of reference of the requests. Another advantage is the ability to exploit parallel processing, making lower response times possible. Since each site can have an arbitrary number of fragments of the data base, there can be any amount of reliability in the event of communication or site failures. One of the major disadvantages of the strategy is the complexity of the system as compared to the other strategies. However, since the amount of duplication is very limited in this particular system, complexity in the design is kept to a minimum.

6.4.2 Data Base Allocation Strategy

As mentioned in the previous section, the smallest fragment in the distributed data base is a relation. In the data base under consideration, relations are of three types: index relations, data relations and directory relations [see Chapter 5]. The directory relation (DR) contains information about all the relations in the data base and serves as a network data directory. The other index relations (IR) contain information about their respective successor-relations in the hierarchy. In the design of the distributed data base allocation procedure, specific data must be assigned to specific sites in the distributed network. In this system, this is done by the following rule: data is assigned to that site where it is collected, i.e.,

the source site. In general, all data relations have only a single copy, at the source site. More than one copy of a data relation can, however, exist in the network, depending on the frequency of reference of any specific data by sites in the network. The directory relation is modified to include the owner site of each data relation in the data base as follows:

Name	Relation	Predecessor	Owner

Fig. 6.1. Modified Data Base Directory Relation

The directory relation serves as a network data directory in the system and is duplicated at every site in the network. Each time a relation is created or destroyed in the system, all copies of this directory are updated; however, although copies of the relation exist at every site, consistency is not of serious concern in the system. This is because of the fact that the directory relation is not a data relation, and only serves as a directory to access other relations in the distributed system. In other words, when a relation is destroyed or created and a query tries to access a copy of

the directory relation before it has been updated, the worst case scenerio is the following: the query returns either the name of a relation that does not actually exist or an error message that the requested relation does not exist. This is not a real problem, as compared to the case when inconsistency occurs in the update of copies of, say, a data relation. In such a case, an erroneous value will be returned without the user ever detecting it.

Apart from the directory relation, there are other index relations that define the hierarchical structure of the data base. Each index relation contains links to all its successor-relations in the data base. Each index relation is linked to its predecessor in the hierarchy, in the directory relation. These index relations have limited duplication in the system. They are duplicated only at those sites where their successor-relations exist. One of the problems with having an arbitrary number of copies occurs when an index relation has to be updated. In such a situation, the location of the different copies of the relation has to be known, to maintain consistency in the data base. To achieve this, the concept of "primary copy"[39] is used. That is, one of the copies of the relation is termed as the primary copy and all other copies become secondary copies. The directory relation maintains information only about the primary copy of the relation and any reference has to be made to the primary copy only. The primary copy, in turn, maintains links or pointers to each of the secondary copies

of the relation. Therefore, by referring to the directory relation and the primary copy, the location of all the copies of an index relation are determined. In order to have consistency in the choice of the primary copy, the first copy of the relation is always marked the primary copy and all further copies of it become secondary copies.

In summary, the directory relation is duplicated at every site in the network, the other index relations are duplicated only at those sites which contain their successor-relations, and the data relations have only one copy at their respective source sites.

6.4.3 The Distributed Environment

The data base environment in a distributed system can be either homogeneous (similar hardware and software at all the sites) or heterogeneous (multiple hardware and/or software systems). Basically, four kinds of environment are possible:

1. a completely homogeneous environment,
2. a heterogeneous hardware cum homogeneous software environment,
3. a homogeneous hardware cum heterogeneous software environment,
4. a completely heterogeneous environment.

In practice, a heterogeneous data base management system environment is generally used only when previously existing autonomous data base systems are being combined

together to form a distributed data base system. In all other situations, a homogeneous environment is preferred because of its reduced complexity.

In geographic applications, as in this specific system, a homogeneous environment is preferred for the following reasons:

- Most geographic data base systems in existence are based on the relational data model [see Chapter 3]. This is because of the fact that the relational model has proved to be more suitable for the representation and manipulation of geographic data than the CODASYL models [see Chapter 2]. Therefore, a homogeneous data base environment based on the relational model of data is the ideal choice.
- The mapping of updates in a heterogeneous environment is still, to a large extent, an unresolved problem. The issue is further complicated when dynamic data translation is required.
- In general, the use of a standard, network-wide user interface and a standard internal form reduces the data translation problem and also the overhead involved in supporting the large number of translation schemes required in the system.
- Because of the large amount of data involved in geographic applications, the use of heterogeneous systems greatly reduces the efficiency for search and retrieval operations on remote data.

A homogeneous environment does not necessarily mean the use of the same DBMS package at all the sites. In such a situation, a common interface can be used in the network. This may be different from that provided by any of the local data base management systems. This entails the conversion of each local schema to the common format. For a network with n sites, this involves only n translation schemes [see Fig. 6.2] as compared to n(n-1) schemes for a completely heterogeneous network.

In summary, a heterogeneous environment greatly complicates the design and implementation of the system and is feasible only when several autonomous data base systems are being combined together to form a distributed network. In most other situations, as in this system, a homogeneous environment is preferred.

6.4.4 Query Processing Strategy

The query processing strategy refers to the steps followed in executing different types of operations in a distributed data base system. The operations in a distributed environment can be divided into two classes:

a. Operations that do not modify the contents of the data base, like search or retrieve.
b. Operations that modify the contents of the data base, like create or update.

The former type does not modify the data base contents and is concerned with factors such as the network

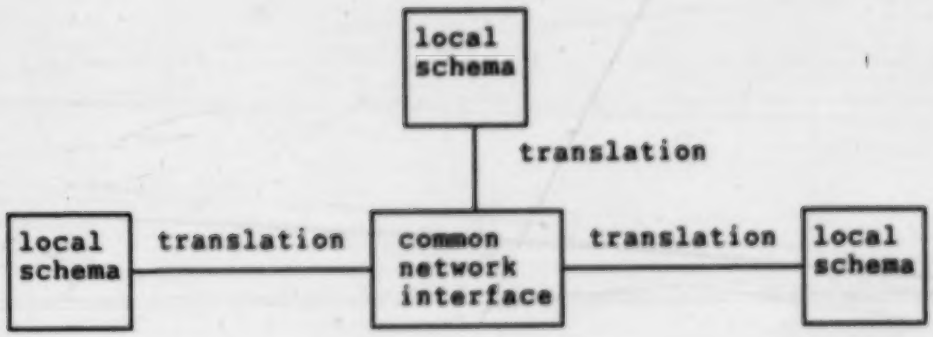


Fig. 6.2. Translation required in a homogeneous environment

configuration, communication protocol, etc. This type of operations is not considered here. The latter type of operations is of three basic types: create, update and destroy.

Only a data relation can be explicitly created by the user. The index relation and the directory relation are created automatically by the system as part of the data relation create process. When a relation is created under a particular site in the hierarchy, all copies of the corresponding index relation (i.e., all copies of the index relation that keeps track of the successor-relations at that particular node in the hierarchy) have to be updated. The network directory, which contains information about all the relations in the distributed system, has to be updated. An algorithm for creation of a relation in the data base is

shown in Algorithm 6.1 (specified at the end of the chapter).

The update operation involves the modification of the contents of a relation. This operation can be performed on any type of relation. The create-relation routine calls this routine to perform the update operations on the relations. The update operation includes add, delete and modify operations on the data. The general procedure for the update operation is shown in Algorithm 6.2 (specified at the end of the chapter).

The third type of operation is destroy-relation. When a relation is destroyed in the distributed data base, all references to it in the data base have to be deleted. In other words, all relations containing reference to the destroyed relation in the data base have to be updated. The steps that have to be taken when a relation is destroyed is given by Algorithm 6.3 (specified at the end of the chapter).

To summarize, there are basically two types of operations: one that modifies the data and one that does not. The latter type can be broadly divided into three classes. All the operations on the data fall into one of these three classes. The algorithms for these operations, presented above, are general in that they do not go into the implementation details which vary from environment to environment and cannot be generalized.

6.5 General Distributed Geographic Data Bases.

In geographic applications, the feasibility of distributing the data base depends primarily on the application itself. As an example, a city planning application might not require a distributed environment whereas a national geological application might favour such an environment. For the latter type of applications, the motivation for distributing the data base has been elaborated in Section 6.1.

The design of a specific distributed geographic data base system in the earlier section has provided some insight into the features of a distributed geographic system. Some of these features are applicable to all distributed geographic systems and can, therefore, be generalized. These include the following:

- Geographic data is, in most cases, voluminous. Therefore in geographically distributed systems, it is not feasible to replicate the actual data. This entails the access of remote data for requests that would otherwise have been satisfied locally in a completely replicated network. However, the disadvantages of having a large amount of redundant data and, therefore, an unusually large memory capacity requirement far outweigh the advantages that can be derived from such a decision.
- A homogeneous data base environment is favoured over a heterogeneous one. Due to the enormous complexities involved in the translation of one schema to another in

a heterogeneous environment, such a system has a large overhead, offers poor performance and becomes very complex in a practical implementation.

- It is a good approach to incorporate in the design a network data directory that maintains information about all the actual data in the distributed system.
- In general, a hybrid data distribution strategy is preferred. In this case, some data, like the directory relation, is duplicated and the remainder is not.
- In distributed geographic applications where the volume of data collected at a data-collection site is high, the data base allocation strategy should be such that data is allocated to the site where it is collected, to avoid large communication cost overheads.

It must be mentioned before closing this section that because of the diversity of geographic applications, it is inadvisable to form a complete set of general guidelines for a distributed geographic data base system. It is, however, reasonable to form a basic set of common design requirements for such a system, as has been done in this section.

6.6 Conclusions

In this chapter, the issue of distributed geographic data bases has been dealt with. The basic motivating factors for distributing geographic data bases have been enumerated. The different distributed data base architectures suitable for geographic applications have been presented. The main

functions of a distributed data base are elaborated. Finally, the design of a distributed geographic data base system has been discussed in detail.

```
procedure create-relation
begin
    Let predecessor index relation be R(p)
    Let no. of copies of R(p) be m-1
    Let the calling site or site be N(i)
    Let the total no. of sites be n
    If (a copy of R(p) does not exist at N(i)) then
    begin
        create a copy of R(p) i.e.,R(p)(m) at N(i)
    end
    create data relation as required
    For j=1 to m do
    begin
        call update-relation( R(p)(j) )
    end
    For k=1 to n do
    begin
        call update-relation(directory relation) at site
        N(k)
    end
end.
```

Algorithm 6.1. Algorithm for the creation of a relation.

```
procedure update-relation( R(i)(j) )
begin
    Let total no. of sites be n
    If ( R(i)(j) is a data relation ) then
    begin
        call concurrency control mechanism
        update R(i)(j)
    end
    else
    begin
        If ( R(i)(j) is an index relation ) then
        begin
            determine primary copy of R(i)(j) i.e.,
            R(i)(P), from network directory
            determine no. of copies of R(i)(j) i.e., m ,
            from R(i)(P)
            For k=1 to m
            begin
                call concurrency control mechanism
                update R(i)(k)
            end
        end
        For k=1 to n
        begin
            call concurrency control mechanism
            for k=1 to n
            update R(k) i.e., copy of network directory
            at site k
        end
    end
end.
```

Algorithm 6.2. Algorithm to update a relation.

```
procedure destroy-relation ( R(i)(j) )
begin
    Let total no. of sites or sites in the system be n
    If ( R(i)(j) is a data relation ) then
    begin
        Let no. of copies of corresponding predecessor
        index relation be p
        check destroy permission of user
        destroy R(i)(j)
        For k=1 to n
        update network directory at site k
        for k=1 to p
        update copy k of the corresponding index
        relation
    end
    else if ( R(i)(j) is an index relation )
    begin
        check destroy permission of user
        If ( R(i)(j) has no successors at site where it
        is to be destroyed ) then
        destroy R(i)(j)
        else return ("cannot be destroyed.")
    end
    else
        If ( R(i)(j) is a copy of network directory )
        then
        return ("cannot be destroyed")
end.
```

Algorithm 6.3. Algorithm to destroy a relation.

CHAPTER 7

CONCLUSIONS

This thesis has dealt with the issue of data base systems for geographic applications. Geographic data base systems embody many of the features of conventional data base systems, and some of their problems too. Geographic data base systems, however, have some features that render them distinctive from conventional data base systems. These include:

a. Large volumes of data and hence, a requirement for an indeterminate amount of storage.
b. The requirement for manipulation of both spatial and aspatial data. These two types of data must be related in some way to facilitate efficient query processing.
c. The need for a set of spatial primitive operators to manipulate geographic data types.
d. The need for specialized input/output equipment to handle geographic data.

Over the years there have been many approaches to designing geographic data base systems. Some of them, like LUIS [19], are special purpose systems in that they are designed for specific applications. There are other general purpose systems that can be used for a variety of applications. General purpose systems are themselves either

specially designed from scratch or extensions to existing conventional data base management systems. One such system, of the latter type, was designed and implemented in this thesis. The system designed in this thesis has some advantages, when compared to other systems like Geo-Quel, that fall under the same class of geographic data base systems. These include advantages in data input, data organization and ease of extension to a distributed environment.

The topic of distributed geographic data base systems has been dealt with, albeit briefly, in this thesis. Proposals have been made for the extension of the data base designed in this thesis to a distributed environment. A whole range of problems, however, have to be researched before concrete proposals for a general purpose distributed geographic data base system can be made and implemented.

In summary, an attempt has been made to consider some of the many issues related to geographic data base systems. One of the prime objectives of this thesis has been to design and implement a geographic data base system using a front-end approach. There are, however, many related issues that need future research. Some of these issues are enumerated in the next section.

7.1 Further Research.

There are many areas of geographic data processing that require further research. They include:

1. A performance evaluation of the front-end and the special-purpose approaches to geographic data base system design. In this thesis, a front-end approach to data base design has been adopted. This should be evaluated with a special-purpose approach. For this, a simulation model of a special-purpose system should be constructed in an operating environment similar to the front-end system. Performance and cost comparisons can then be made between the two approaches to determine their relative cost effectiveness.
2. Implementation of a link between data base and file system modules.

   The link between the data base module and the file system module is not in working order. This is mainly due to the fact that the file system is incomplete in its implementation. This link will facilitate the storage of data relations as files in the UNIX file system, which can be called from the file system module of the geographic system. Further work is required to make this possible, and in working order.
3. The application of distributed data processing systems to geographic systems. The area of distributed geographic data processing system is a relatively unexplored one. There are a whole range of problems in

this area that need to be studied and researched.

4. The design of computer architectures for geographic information systems.

   Geographic applications require a fast response time on large volumes of geographic data. Computer architectures can be designed to meet, at least partially, this requirement. Image parallelism can be exploited to enhance performance and cost-effectiveness. The recent developments in VLSI technology make it feasible to produce high performance architectures for geographic applications.

The issues listed above are only a few of the several that need to be considered by researchers in the future. It is hoped that in the course of time, efficient solutions can be found to many of these problems that will, eventually, make geographic data base systems much more efficient, cost-effective and easier to design and manipulate.

REFERENCES

1. F. Antonacci, L. Bartolo, P. DellOrco, V. Spadavecchia, "AQL - A Relational Data Base Management System and its Geographical Applications", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

2. R.R. Berman, M. Stonebraker, "GEO-QUEL: A System for the Manipulation and Display of Geographic Data", *Proceedings of ACM Siggraph 77*, 1977.

3. R.F. Boyce, "Specifying queries as relational expressions", *Data Base Management*, North-Holland Publ., 1974.

4. C.R. Carlson, "ADAM: a general purpose algebraic data access and manipulation sublanguage", *Dept. of Computer Science*, Northwestern University, Illinois, 1975.

5. G.A. Champine, *Distributed Computer Systems*, North-Holland, 1980.

6. S.K. Chang, N. Donato, B.H. McCormick, J. Reuss, R. Rocchetti, "A Relational Data Base System for pictures", *Proceedings of the workshop on data description and management*, 1977.

7. CODASYL, "CODASYL Data Base Task Group", *April 71 Report, ACM, New York*, 1971.

8. E.F. Codd, "Relational completeness of data base sublanguages", *Data Base Systems*, Prentice Hall, 1972

9. G.S. Dutton, W.G. Nisen, "The Expanding Realm of Computer Cartography", *Datamation*, Vol.24, pages 134-142, 1978.

10. R. Epstein, "Creating and Maintaining a Data Base

using Ingres", *Memorandum No. UCB/ERL M77/71*, Electronic Research Lab., Univ. of California, 1975.

11. G. Held, M. Stonebraker, "Networks, Hierarchies and Relations in Data Base Management Systems", *Memorandum No. ERL-M504*, Electronics Research Lab., Univ. of California, 1975.

12. D.A. Jardine (Ed.), *Data Base Management Systems*, North-Holland Publ., 1974.

13. Juan Fco. Corona Burgueno, "A Geographical Data Base", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

14. S.K. Fu, N.S. Chang, "A Relational Data Base System for Images", *Pictorial Information Systems*, Lecture Notes in Computer Science, No.80, 1980.

15. R.A. Lorie, J.L. Becerril, R. Casajuana, "GSYSR: A Relational Data Base Interface for Graphics", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

16. R.M. Males, W.E. Gates, "ADAPT: A Digital Terrain Model-Based Geographic Information System", *Cartographic and Statistical Data Bases and Mapping Software*, Harvard Library of Computer graphics, 1980 Mapping Collection.

17. P.E. Mantey, E.D. Carlson, "Integrated Geographic Data Bases: The GADS Experience", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

18. D.M. McKeown Jr, "MAPS: The Organization of a Spatial Data Base System using Imagery, Terrain and Map Data", *Dept. of Computer Science*, Carnegie-Mellon University, July 1983.

19. C. Meade, "LUIS: An Interactive Graphics System used for Data Base Management in a Regional Planning Environment", *Urban, Regional and State Applications*, Harvard Library of Computer Graphics, 1979 Mapping Collection.

20. A.S. Micheals, B. Mittman, "A Comparison of the Relational and Codasyl approaches to Data Base Management", *Computing Surveys*, Vol.8, No.1, March 1976.

21. M.S. Monmonier, *Computer-Assisted Cartography: Principles and Prospects*, Prentice Hall, 1982

22. G. Nagy, S. Wagle, "Geographic Data Processing", *ACM Computing Surveys*, Vol.11, No.2, pages 139-181, 1979.

23. A. Olsson, "A Spatial Information System: A Pilot Study", *Central Board of Real Estate Data*, Sunbyberg, Sweden; FRIS C:3, May 1971.

24. J.L. Parker, "Information Retrieval with Large-Scale Geographic Data Bases", *Report No.1*, Dept. of Computer Science, Univ. of British Columbia, June 1971

25. H. Schmutz, "The Integrated Data Analysis and Management System for Pictorial Applications", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

26. B. Smedley, B. Aldred, "Problems with geo-data", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

27. M. Stonebraker, C. Williams, A. Go, "An Approach to Implementing a GEO-DATA System", *ACM-SIGMOD Workshop on Data Base in Interactive Design*, Waterloo, Ontario, September 1975.

28. Y. Takao, S. Itoh, J. Iisaka, "An Image-oriented Data Base System", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

29. J.D. Ullman, *Principles of Data Base Systems*, Second Edition, Computer Science Press, 1982.

30. W.J. Waghorn, "The DDL as an industry standard", *Data Base Description*, North-Holland, 1975.

31. F. Palermo, D. Weller, "Some Data Base Requirements for Pictorial Applications", *Data Base Techniques for Pictorial Applications*, Springer-Verlag 1980.

32. D. White, "The ODYSSEY System for Display and Analysis of Geographic Information", *Cartographic and Statistical Data Bases and Mapping Software*, Harvard Library of Computer Graphics, 1980 Mapping Collection.

33. E.S. Page, L.B. Wilson, "Information Representation and Manipulation in a Computer", *2nd. edition*, Cambridge Univ. Press, 1978.

34. E. Wong, M. Stonebraker, P. Kreps, G. Held, "The Design and Implementation of INGRES", *ACM Transactions on Data Base Systems*, Vol.1, No.3, September 1976.

35. ESRI, USA, "ARC/INFO:Computer Mapping and Geographic Information System", *Product Information Report*, 1984.

36. G. Tang, "A Logical Data Organization for the Integrated Data Base of Pictures and Alphanumeric Data", *Proc. IEEE* , 1980.

37. S. Kubo, "ALIS: A Geographical Information System for Urban Research", *IEEE Graphics*, Vol.3, 1983.

38. C.H. Helmers, W.A. Davis, "CIAD File System Kernel: An Overview", *Dept. of Computer Science, University of Alberta*, 25 February 1984.

39. M.Stonebraker, "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES", "IEEE Transactions on Software Engineering", *Vol.1, SE-5, No.3*, May 1979.

40. D.M.Ritchie, K.Thompson, "The UNIX Time-Sharing System", *Communications of the ACM*, 17, No.7, pp. 365-375, July 1974.

APPENDIX I

QUERY ROUTINES FOR THE GEOGRAPHIC DATA BASE

The query routines for the geographic data base implemented include:

1. Print relation
2. Attribute relation
3. Delete relation where { qualification }
4. Create relation
5. RDelete relation where { qualification }
6. Destroy relation
7. Getattr attribute(s) of relation
8. Getpath { of a relation }
9. Getr relation(entity-name) of relation1(entity-name) = relation2(entity-name)
10. Getar attribute(s) of relation(entity-name)
11. Getarp attribute(s) of relation(entity-name) for relation1(entity-name) = relation2(entity-name)
12. Locate relation(entity-name)
13. Hiersrch relation(entity-name)
14. Getrel relation(entity-name)
15. Getpath { relation(entity-name) }

    Find {attributes} of {relation} where {qual1} {op} {qual2}

LOCATE

Name: Locate - to locate a relation in the data base, given the name of the attribute.

Synopsis: Locate n (where n is the name of the entity).

Description: This is a command routine which locates the name of the relation corresponding to a specified entity. This function, when called, starts up the INGRES DBMS and invokes the appropriate data base in it. It then searches the index relation (in this case the relation RELATIONS which contains all the attribute names and their corresponding relation names) for a match and retrieves all the tuples that satisfy the match.

If a match is not found in the first pass, the function informs the user of the same and then makes a second pass for a sub-string(partial) match. If no match is found in the second pass too, the function prints out a list of the valid relations. The user can then select a valid entity name from this list.

The function also checks for syntax errors in the function specification. For increased efficiency, the function does not start up the INGRES DBMS whenever a syntax error is present in the function specification.

Files:
/u1/grad/mohan/thesis/PROG/locfin.q
/u1/grad/mohan/thesis/PROG/locfin.c
/u1/grad/mohan/thesis/PROG/Locate

PRINT

Name: Print - to print out a specified relation.

Synopsis: Print n (where n is the name of a relation).

Description: This is a UNIX function which prints out the specified relation. This function, when called, starts up the INGRES DBMS and invokes the appropriate data base in it. It then searches the data base for the appropriate relation and prints it out.

If no relation in the data base satisfies the specified relation name the function prints out the appropriate message to the user.

The function also checks for syntax errors in the function specification. The function does not start up the INGRES DBMS whenever a syntax error is present in the function specification.This reduces response time and increases the efficiency of operation.

Files:
/u1/grad/mohan/thesis/PROG/print.q
/u1/grad/mohan/thesis/PROG/print.c
/u1/grad/mohan/thesis/PROG/Print

ATTRIBUTE

Name: Attribute - to print out the attributes of a specified relation and the format type of each of them.

Synopsis: Attribute n (where n is the name of the relation).

Description: This is a UNIX function which prints out the attributes of a specified relation. This function, when called, starts up the INGRES DBMS and invokes the appropriate data base in it. It then searches the ingres system relations for a match and retrieves the names of all the attributes that that satisfy the qualification.

If no attributes satisfy the match (i.e., the specified relation does not exist) the function prints out the appropriate message to the user.

The function also checks for syntax errors in the function specification. To improve efficiency and speed of operation, the function does not start up the INGRES DBMS whenever a syntax error is present in the function specification.

Files:
/u1/grad/mohan/thesis/PROG/attr.q
/u1/grad/mohan/thesis/PROG/attr.c
/u1/grad/mohan/thesis/PROG/Attribute

GETREL

Name: Getrel - to retrieve relation names corresponding to a specified entity, store them in an array and pass on the array pointer to the calling routine.

Description: This C-QUEL function retrieves relation names corresponding to a specified entity, stores them in an array and passes on the pointer to the array to the calling program. This function, when called, starts up the INGRES DBMS and invokes the appropriate data base in it. It then searches the index relation for a match and retrieves the names of all the relations that that correspond to the specified entity.

If no relation satisfies the match the function prints out the appropriate message to the user.

The function also checks for syntax errors in the function specification.

The retrieved relation names are stored in array and pointers to this array are passed to the calling program.

Files: /u1/grad/mohan/thesis/PROG/getrel.q
/u1/grad/mohan/thesis/PROG/getrel.c

DELETE

Name: Delete - to delete tuples in relations in the data base, given the name of the relation and the qualifications.

Synopsis: Delete from (n) records where (attribute1) (operator) (attribute2).
n is the name of an entity).

Description: This command routine deletes records from relations in the data base that match specified qualifications. The relations are specified as entities (i.e., their English language equivalents). The routine retrieves relations that correspond to the specified attributes and deletes from them records that match specified qualifications. The operators allowed in the specification of the qualifications are: GT(greater than), GE(greater or equal), EQ(equal), LE(less than or equal), LT(less than).
The routine also checks for syntax errors.

Files: /u1/grad/mohan/thesis/PROG/delete.q
/u1/grad/mohan/thesis/PROG/delete.c
/u1/grad/mohan/thesis/PROG/Delete

CREATE

Name: Create - to create a new relation in the data base.

Synopsis: Create (relation) (entity-name) (predecessor) ((attribute-format)).

Description: This command creates relations in the data base. It also updates the specified predecessor relation. The directory relation (the data base directory) is automatically updated when a relation is created. The created relation can contain any number of attributes. Any of the legal INGRES formats can be used in the Create command.

The Create command builds up a string from the specified arguments and then uses IIwrite to create a relation in the data base.

Files:
/u1/grad/mohan/thesis/PROG/create.q
/u1/grad/mohan/thesis/PROG/create.c
/u1/grad/mohan/thesis/PROG/Create

DESTROY

Name: Destroy - to destroy specified relations in the data base.

Synopsis: Destroy n (where n is the name of the relation specified as an attribute).

Description: This query command destroys specified existing relations from the data base. A relation can be destroyed only by its owner. The routine also deletes corresponding entries from the index relation and also from its predecessor. The predecessor is determined by calling another C-QUEL routine.
The routine also checks for syntax errors in the command specification.

Files: /u1/grad/mohan/thesis/PROG/destroy.q
/u1/grad/mohan/thesis/PROG/destroy.c
/u1/grad/mohan/thesis/PROG/Destroy

RDELETE

Name: RDelete - to delete tuples from relations in the data base that match specified qualifications.

Synopsis: RDelete from (relation) where (attribute1) (operator) (attribute2).

Description: This routine is similar to the "Delete" routine. Records or tuples that match specified qualifications are deleted from desired relations in the data base. However, in this routine, the actual name of the relation is specified by the user instead of its attribute(i.e., its English language equivalent).

Files: /u1/grad/mohan/thesis/PROG/deleter.q
/u1/grad/mohan/thesis/PROG/deleter.c
/u1/grad/mohan/thesis/PROG/RDelete

Getar (GET ATTRIBUTE)

Name: Getar - to retrieve specific attribute(s) of relations.

Synopsis: Getar (attribute1 ..attributeN) of (relation). Relation is specified as its entity-name (i.e., its English language equivalent).

Description: This query command retrieves information about specific attributes of specified relations in the data base. The relations are specified in the command as their entity-names. The routine retrieves relation name(s) that correspond to the specified entity, does a character-conversion of the retrieved name (from upper case to lower case), retrieves the required information from the specified relation and prints them out. Like all other routines, this routine prints out detailed error messages.

Files: /u1/grad/mohan/thesis/PROG/getAr.q
/u1/grad/mohan/thesis/PROG/getAr.c
/u1/grad/mohan/thesis/PROG/Getar

GETATTR (GET ATTRIBUTE of relation)

Name: Getattr - to retrieve specified attribute(s) of relations.

Synopsis: Getattr (attribute1 ..attributeN) of (relation).

Description: This command routine displays the contents of the specified attributes in the specified relation on the standard output. It is similar to the command routine *Getar*. This routine also checks for syntax errors in the command specification.

Files:
/u1/grad/mohan/thesis/PROG/getA.q
/u1/grad/mohan/thesis/PROG/getA.c
/u1/grad/mohan/thesis/PROG/Getattr

GETR (Get Relation)

Name: Getr - to retrieve a relation that satisfies the specified qualification.

Synopsis: Getr (relation) of (relation1) = (relation2).
All the relations are specified as their entity-names (i.e., their English language equivalents).

Description: This command is used to retrieve relations that match specific user-defined qualifications. The qualifications in this command refer to the specification of the path (in the hierarchy) that should include the specified relation. This command calls another function to locate the entire path of the relation to be retrieved and checks whether this located path includes relation1 and relation2, where relation2 is a successor of relation1 in the hierarchy. The routine calls yet another function to identify the relation names that correspond to the specified entity-names in the command.
To improve efficiency and speed of operation, the function does not start up the INGRES DBMS whenever a syntax error is present in the function specification.

Files: /u1/grad/mohan/thesis/PROG/getR.q
/u1/grad/mohan/thesis/PROG/getR.c
/u1/grad/mohan/thesis/PROG/Getr

GETARP

Name: Getarp - to get the attributes of a relation that match the specified path-qualifications.

Synopsis: Getarp (attribute1 ..attributeN) of (relation) for (relation1) = (relation2) .

Description: This command routine combines the features of two of the earlier mentioned commands (Getattr and Getr). It retrieves the attribute(s) of the specified relation that matches the required path-qualifications. Path-qualifications refer to the specification of the path (in the hierarchy) that should include the specified relation. It also does extensive error checking and prints out detailed error messages on the standard output.

Files:
/u1/grad/mohan/thesis/PROG/quel:gAR.q
/u1/grad/mohan/thesis/PROG/quel:gAR.c
/u1/grad/mohan/thesis/Getarp

SEARCH

Name: Search - to retrieve the relation name(s) that correspond to a specified attribute (entity) and identify the path in the hierarchy of each of them.

Synopsis: Search n (where n is the name of the entity).

Description: This routine retrieves all the relation names that correspond to a specified entity name and calls a function to identify the path in the hierarchy of each of the retrieved relations. It also does the required error checking.

Files:
/u1/grad/mohan/thesis/PROG/hiersrch.q
/u1/grad/mohan/thesis/PROG/hiersrch.c
/u1/grad/mohan/thesis/PROG/Search

FIND

Name: Find - to retrieve the attributes of the tuples of the specified relation that satisfies the specified qualifications.

Synopsis: Find {attributes} of {relation(entity-name)} where {qualification1} {operator} {qualification2}.

Description: This routine retrieves the specified attributes or domains of the tuples in the relation that satisfy the specified qualifications. The relation is specified as its entity-name. The corresponding relation name is derived from the data base directories. The operators can be EQ(equal to), GE(greater or equal to), GT (greater than), LT(less than), or LE(less than or equal to). It also does the necessary error checking.

Files: /u1/grad/mohan/thesis/PROG/find.q
/u1/grad/mohan/thesis/PROG/find.c
/u1/grad/mohan/thesis/PROG/Find

Other Functions:

Procattr.q:- This routine check the validity of any specified attribute. It basically checks whether the specified relation has the specified attribute or not. It also checks for errors in the command.

Getpath.q:- This is the function that identifies the path of any specified relation in the hierarchically stored data base. The path is the list of all nodes, in a sequence, from the specified relation to the root. The lists are stored in three dimensional arrays and pointers to their location are passed to the calling program.

APPENDIX II

GEOGRAPHIC NAME DATA BASE

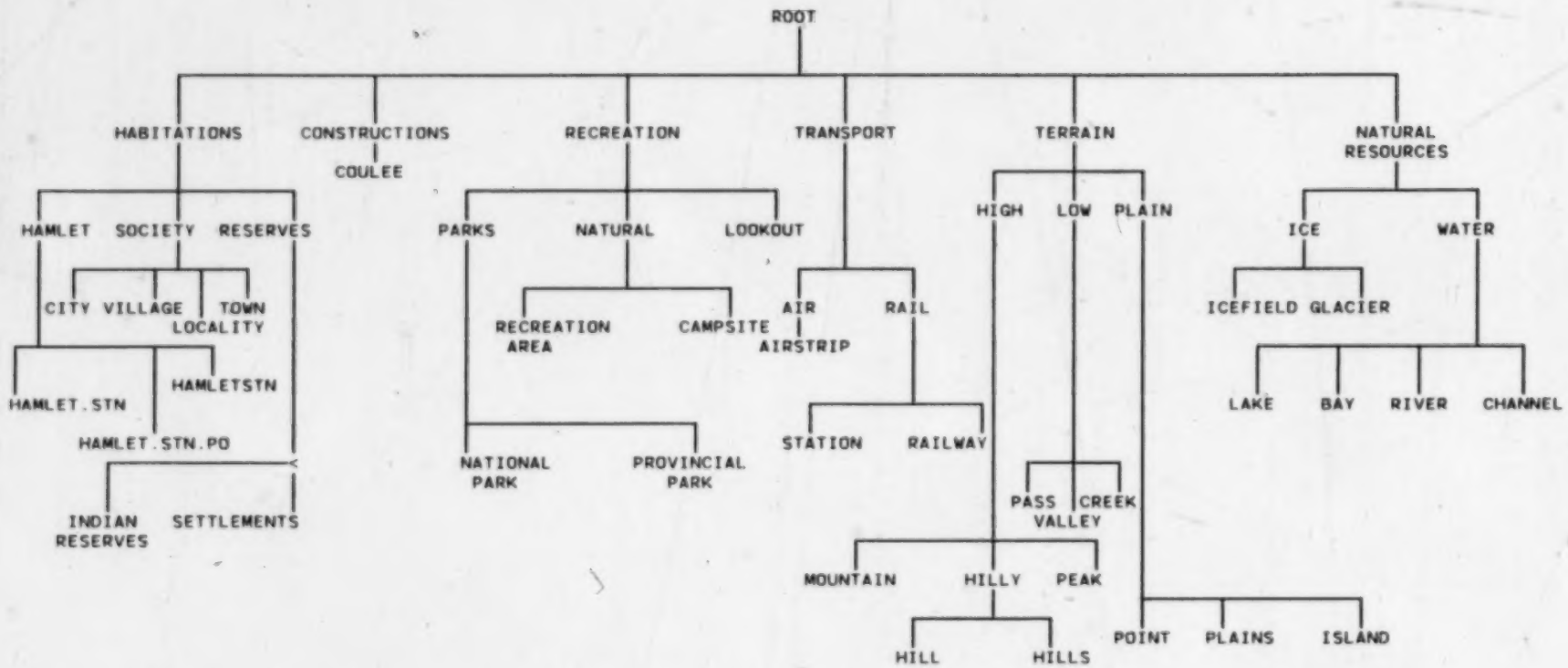
Geographic Name File format

Tape Format

Density:	1600 bpi
Length:	2400 feet
Record Format:	Fixed Block
Record Size:	170 bytes
Block Size:	10 records
Labels:	IBM standard
Format:	EBCDIC Characters

Column	Field	Contents
001-005	Number	Record Sequence Number
006-010		Zeros
011-038	Name	Geographic Place Name
039-040		Blanks
041-059	Feature	Feature type
060-060	Flag	Short List Flag
061-090	Description	Description of place name
091-093	Source	Original source of data
094-110	Misc	4 miscellaneous fields
111-111	Surveyed	Not used
112-113	Part	Part section (i.e., NW, SE, E, etc.)
114-115	W.Mer	West of meridian (coded as W4, W5, or W6)
116-117		Blanks
118-122	Lot Mapsheet	1:50,000 NTS map sheet number
123-124	Legal Sub	Legal subdivision (or zeros or blanks)
125-126	Section	Section number
127-129	Township	Township number
130-131	Range	Range number
132-135	Pub Date	Date name was published
136-148	Position From	Not used
149-161	Position	Lat/Long position subdivided as:
149-150	Lat 1	Latitude - in degrees
151-152	Lat 2	Latitude - in minutes
153-154	Lat 3	Latitude - in seconds (set to zero)
155-157	Long 1	Longitude - in degrees
158-159	Long 2	Longitude - in minutes
160-161	Long 3	Longitude - in seconds (set to zero)
162-168		Blanks
169-170	Sort Code	?????????

APPENDIX III



GEOGRAPHIC DATA BASE STRUCTURE

APPENDIX IV

SAMPLE QUERY RESULTS

1. Locate abc def

```
wrong number of parameters
correct usage: locate  n
```

2. Locate rail

Attribute	Relation
rail	RAIL

3. Locate river

Attribute	Relation
river	RIVER
river	RIVER2

4. Locate hill

Attribute	Relation
hill	HILL

5. Locate hil

Attribute	Relation

```
No relation  matches the required characteristics
Searching for a sub-string(partial) match.....
```

hilly	HILLY
hill	HILL
high	HIGH
hills	HILLS

6. Locate mohan

Attribute	Relation

```
No relation  matches the required characteristics
Searching for a sub-string(partial) match.....

No partial match exists.
```

The valid relations are (press return):

relations relation

attribute	relation	predecessor
parks	PARKS	RECREATION
town	TOWN	SOCIETY
Indian.reserves	INDRES	RESERVES
constructions	CONST	ROOT
hilly	HILLY	HIGH
point	POINT	PLAIN
recreation	RECREATION	ROOT
village	VILLAGE	SOCIETY
natural.resources	NATRES	ROOT
campsite	CAMPSITE	NATURAL
city	CITY	SOCIETY
hamlet	HAMLET	HABITATION
hamlet.stn	HAMLETSTN	HAMLET
peak	PEAK	HIGH
reserves	RESERVES	HABITATION
reservoir	RES	CONST
river	RIVER	WATER
settlements	SETTLEM	RESERVES
society	SOCIETY	HABITATION
terrain	TERRAIN	ROOT
water	WATER	NATRES
river	RIVER2	ICEFIELD
hamletstn	HAMLETST	HAMLET
hill	HILL	HILLY
lake	LAKE	WATER
recreation.area	RECAREA	NATURAL
bay	BAY	WATER
coulee	COULEE	CONST
glacier	GLACIER	ICE
habitations	HABITATION	ROOT
icefield	ICEFIELD	ICE
plains	PLAINS	PLAIN
national.park	NATPARK	PARKS
air	AIR	TRANSPORT
cabin	CABIN	CONST
high	HIGH	TERRAIN
hills	HILLS	HILLY
locality	LOCALITY	RESERVES
natural	NATURAL	RECREATION
provincial.park	PROVPARK	PARKS
channel	CHANNEL	WATER
hamlet.stn.po	HAMLETPO	HAMLET
island	ISLAND	PLAIN
low	LOW	TERRAIN
valley	VALLEY	LOW
airstrip	AIRSTRIP	AIR

plain	PLAIN	TERRAIN
rail	RAIL	TRANSPORT
root	ROOT	
creek	CREEK	LOW
lookout	LOOKOUT	RECREATION
mountain	MOUNTAIN	HIGH
pass	PASS	LOW
range	RANGE	CONST
station	STATION	RAIL
ice	ICE	NATRES
railway	RAILWAY	RAIL
transport	TRANSPORT	ROOT

7. Print abc def

Wrong number of arguments
Correct usage: print n

8. Print natural

natural relation

name	relation
recreation area	RECAREA
campsite	CAMPSITE

9. Attribute relations

The attributes of relations are:
1. attribute c18
2. relation c15
3. predecessor c15

10. Attribute abc def

Wrong number of parameters
Correct usage: Attribute n

11. Attribute palat

Relation palat does not exist

12. Print society

society relation

name	relation
town	TOWN
village	VILLAGE
city	CITY

13. Print relations

relations relation

attribute	relation	predecessor
parks	PARKS	RECREATION
town	TOWN	SOCIETY
Indian.reserves	INDRES	RESERVES
constructions	CONST	ROOT
hilly	HILLY	HIGH
point	POINT	PLAIN
recreation	RECREATION	ROOT
village	VILLAGE	SOCIETY
natural.resources	NATRES	ROOT
campsite	CAMPSITE	NATURAL
city	CITY	SOCIETY
hamlet	HAMLET	HABITATION
hamlet.stn	HAMLETSTN	HAMLET
peak	PEAK	HIGH
reserves	RESERVES	HABITATION
reservoir	RES	CONST
river	RIVER	WATER
settlements	SETTLEM	RESERVES
society	SOCIETY	HABITATION
terrain	TERRAIN	ROOT
water	WATER	NATRES
river	RIVER2	ICEFIELD
hamletstn	HAMLETST	HAMLET
hill	HILL	HILLY
lake	LAKE	WATER
recreation.area	RECAREA	NATURAL
bay	BAY	WATER
coulee	COULEE	CONST
glacier	GLACIER	ICE
habitations	HABITATION	ROOT
icefield	ICEFIELD	ICE
plains	PLAINS	PLAIN
national.park	NATPARK	PARKS
air	AIR	TRANSPORT
cabin	CABIN	CONST
high	HIGH	TERRAIN
hills	HILLS	HILLY
locality	LOCALITY	RESERVES
natural	NATURAL	RECREATION
provincial.park	PROVPARK	PARKS
channel	CHANNEL	WATER
hamlet.stn.po	HAMLETPO	HAMLET
island	ISLAND	PLAIN
low	LOW	TERRAIN
valley	VALLEY	LOW
airstrip	AIRSTRIP	AIR

plain	PLAIN	TERRAIN
rail	RAIL	TRANSPORT
root	ROOT	
creek	CREEK	LOW
lookout	LOOKOUT	RECREATION
mountain	MOUNTAIN	HIGH
pass	PASS	LOW
range	RANGE	CONST
station	STATION	RAIL
ice	ICE	NATRES
railway	RAILWAY	RAIL
transport	TRANSPORT	ROOT
palatt	PALAT	hilly
wildlife	WILD	natural
all	wild	

14.Create mohan MOHANP SOCIETY name=c18,description=c10

```
create MOHANP(
create MOHANP( name=c18,description=c10
qual is create MOHANP( name=c18,description=c10)
```

15. Print SOCIETY

society relation

name	relation
town	TOWN
village	VILLAGE
city	CITY
mohan	MOHANP

16. Print relations

relations relation

attribute	relation	predecessor
parks	PARKS	RECREATION
town	TOWN	SOCIETY
Indian.reserves	INDRES	RESERVES
constructions	CONST	ROOT
hilly	HILLY	HIGH
point	POINT	PLAIN
recreation	RECREATION	ROOT
village	VILLAGE	SOCIETY
natural.resources	NATRES	ROOT
campsite	CAMPSITE	NATURAL
city	CITY	SOCIETY
hamlet	HAMLET	HABITATION
hamlet.stn	HAMLETSTN	HAMLET
peak	PEAK	HIGH

reserves	RESERVES	HABITATION
reservoir	RES	CONST
river	RIVER	WATER
settlements	SETTLEM	RESERVES
society	SOCIETY	HABITATION
terrain	TERRAIN	ROOT
water	WATER	NATRES
river	RIVER2	ICEFIELD
hamletstn	HAMLETST	HAMLET
hill	HILL	HILLY
lake	LAKE	WATER
recreation.area	RECAREA	NATURAL
bay	BAY	WATER
coulee	COULEE	CONST
glacier	GLACIER	ICE
habitations	HABITATION	ROOT
icefield	ICEFIELD	ICE
plains	PLAINS	PLAIN
national.park	NATPARK	PARKS
air	AIR	TRANSPORT
cabin	CABIN	CONST
high	HIGH	TERRAIN
hills	HILLS	HILLY
locality	LOCALITY	RESERVES
natural	NATURAL	RECREATION
provincial.park	PROVPARK	PARKS
channel	CHANNEL	WATER
hamlet.stn.po	HAMLETPO	HAMLET
island	ISLAND	PLAIN
low	LOW	TERRAIN
valley	VALLEY	LOW
airstrip	AIRSTRIP	AIR
plain	PLAIN	TERRAIN
rail	RAIL	TRANSPORT
root	ROOT	
creek	CREEK	LOW
lookout	LOOKOUT	RECREATION
mountain	MOUNTAIN	HIGH
pass	PASS	LOW
range	RANGE	CONST
station	STATION	RAIL
ice	ICE	NATRES
railway	RAILWAY	RAIL
transport	TRANSPORT	ROOT
palatt	PALAT	hilly
wildlife	WILD	natural
all	wild	
mohan	MOHANP	society

17. Create xyz

```
create xyz(
qual is create xyz()
INGRES ERROR:line 1,Syntax error on ')',correct syntax is:
CREATE relname (domname1 = format(, domname2 = format))
```

18. Delete from xyz

```
Command specification error
Argument(s) missing from command specification
```

19. Delete from palat records where name EQ "mohan"

```
No relation matches specified attribute palat
```

20. Delete from river records where name EQ athabasca

```
relation is RIVER
relation is RIVER2
More than one relation matches attribute river
```

21. Delete from society records where name EN mohan

```
relation is SOCIETY
Error: Invalid operator EN
```

22. Delete from society records where name EQ mohan

```
relation is SOCIETY
```

23.Print society

society relation

name	relation
town	TOWN
village	VILLAGE
city	CITY

24. Print relations

relations relation

attribute	relation	predecessor
parks	PARKS	RECREATION
town	TOWN	SOCIETY
Indian.reserves	INDRES	RESERVES
constructions	CONST	ROOT
hilly	HILLY	HIGH

point	POINT	PLAIN
recreation	RECREATION	ROOT
village	VILLAGE	SOCIETY
natural.resources	NATRES	ROOT
campsite	CAMPSITE	NATURAL
city	CITY	SOCIETY
hamlet	HAMLET	HABITATION
hamlet.stn	HAMLETSTN	HAMLET
peak	PEAK	HIGH
reserves	RESERVES	HABITATION
reservoir	RES	CONST
river	RIVER	WATER
settlements	SETTLEM	RESERVES
society	SOCIETY	HABITATION
terrain	TERRAIN	ROOT
water	WATER	NATRES
river	RIVER2	ICEFIELD
hamletstn	HAMLETST	HAMLET
hill	HILL	HILLY
lake	LAKE	WATER
recreation.area	RECAREA	NATURAL
bay	BAY	WATER
coulee	COULEE	CONST
glacier	GLACIER	ICE
habitations	HABITATION	ROOT
icefield	ICEFIELD	ICE
plains	PLAINS	PLAIN
national.park	NATPARK	PARKS
air	AIR	TRANSPORT
cabin	CABIN	CONST
high	HIGH	TERRAIN
hills	HILLS	HILLY
locality	LOCALITY	RESERVES
natural	NATURAL	RECREATION
provincial.park	PROVPARK	PARKS
channel	CHANNEL	WATER
hamlet.stn.po	HAMLETPO	HAMLET
island	ISLAND	PLAIN
low	LOW	TERRAIN
valley	VALLEY	LOW
airstrip	AIRSTRIP	AIR
plain	PLAIN	TERRAIN
rail	RAIL	TRANSPORT
root	ROOT	
creek	CREEK	LOW
lookout	LOOKOUT	RECREATION
mountain	MOUNTAIN	HIGH
pass	PASS	LOW
range	RANGE	CONST
station	STATION	RAIL
ice	ICE	NATRES
railway	RAILWAY	RAIL
transport	TRANSPORT	ROOT

mohan	MOHANP	society

25. Print mohanp

mohanp relation

name	descriptio

26. Destroy mohan

relation is MOHANP
 dum is SOCIETY
 dum2 is society

27.Print mohanp

5001: PRINT: bad relation name mohanp

28. Print relations

relations relation

attribute	relation	predecessor
parks	PARKS	RECREATION
town	TOWN	SOCIETY
Indian.reserves	INDRES	RESERVES
constructions	CONST	ROOT
hilly	HILLY	HIGH
point	POINT	PLAIN
recreation	RECREATION	ROOT
village	VILLAGE	SOCIETY
natural.resources	NATRES	ROOT
campsite	CAMPSITE	NATURAL
city	CITY	SOCIETY
hamlet	HAMLET	HABITATION
hamlet.stn	HAMLETSTN	HAMLET
peak	PEAK	HIGH
reserves	RESERVES	HABITATION
reservoir	RES	CONST
river	RIVER	WATER
settlements	SETTLEM	RESERVES
society	SOCIETY	HABITATION
terrain	TERRAIN	ROOT
water	WATER	NATRES
river	RIVER2	ICEFIELD
hamletstn	HAMLETST	HAMLET
hill	HILL	HILLY
lake	LAKE	WATER
recreation.area	RECAREA	NATURAL
bay	BAY	WATER
coulee	COULEE	CONST

glacier	GLACIER	ICE
habitations	HABITATION	ROOT
icefield	ICEFIELD	ICE
plains	PLAINS	PLAIN
national.park	NATPARK	PARKS
air	AIR	TRANSPORT
cabin	CABIN	CONST
high	HIGH	TERRAIN
hills	HILLS	HILLY
locality	LOCALITY	RESERVES
natural	NATURAL	RECREATION
provincial.park	PROVPARK	PARKS
channel	CHANNEL	WATER
hamlet.stn.po	HAMLETPO	HAMLET
island	ISLAND	PLAIN
low	LOW	TERRAIN
valley	VALLEY	LOW
airstrip	AIRSTRIP	AIR
plain	PLAIN	TERRAIN
rail	RAIL	TRANSPORT
root	ROOT	
creek	CREEK	LOW
lookout	LOOKOUT	RECREATION
mountain	MOUNTAIN	HIGH
pass	PASS	LOW
range	RANGE	CONST
station	STATION	RAIL
ice	ICE	NATRES
railway	RAILWAY	RAIL
transport	TRANSPORT	ROOT

29. Search abc def

Error in command specification.
Correct usage: Search n

30. Search palat

No relation matches specified attribute palat

31. Search hill

Relations are:

HILL

The required path of HILL in the hierarchy is :

HILL HILLY HIGH TERRAIN ROOT

```
32. Search river

Relations are:
--------------

RIVER
RIVER2

 The required path of RIVER in the hierarchy is :

RIVER          WATER          NATRES          ROOT

 The required path of RIVER2 in the hierarchy is :

RIVER2      ICEFIELD      ICE      NATRES      ROOT

33. Search root

Relations are:
--------------

ROOT
 The required path of ROOT in the hierarchy is :

ROOT

34. Getattr name

ERROR in command specification
Correct usage: Getattr attribute(s) of relation

35. Getattr name of AIRSTRIP

ELK POINT
FT MACLEOD
ROCKYFORD
FOX CREEK
DONNELLY
MERIDIAN
PEACE RIVER
MOBIL OIL
STEEN RIVER

36.Getattr name source of RAILWAY

ABILENE JUNCTION                CNR
SWAN LANDING                    CNR
KNIGHT                          GAZ

37.Getar name of reln
No relation matches specified attribute reln
```

```
38.Getar name of river

relation is RIVER
relation is RIVER2
More than one  relation matches specified attribute river
relation is BAY

39.Getar name of bay

FRASER BAY
RUSSEN HOLT BAY

40.RDelete from xyz records where name EQ mohan

xyz: No such relation exists in the data base

41.RDelete from HILLY

Command specification error
Argument(s) missing from command specification


42.RDelete from HILLY records where xyz EQ mohan

INGRES ERROR: Attribute 'xyz' not in relation 'hilly'


43.RDelete from HILLY records where name EE mohan

Error: Invalid operator EE

44. Print hilly

hilly relation
```

name	relation
mohan	MOHAN
hills	HILLS
hill	HILL

```
45. RDelete from HILLY records where name EQ mohan

46. Print hilly

hilly relation
```

name	relation
hills	HILLS
hill	HILL

47.Getr a

ERROR in command specification:
Correct format:Getr relation of relation1•relation2

48. Getr name of high = hilly

relation is HIGH
relation is HILLY
No relation matches specified attribute name

49.Getr hill of high = name

relation is HILL
relation is HIGH
No relation matches specified attribute•name

50.Getr station of transport = rail

relation is STATION
relation is TRANSPORT
relation is RAIL
relations are: STATION
The required path of station in the hierarchy is :

STATION RAIL TRANSPORT ROOT
station relation

recno	name	feature	description	source
06482	MITFORD	STATION	CPR	GAZ
02754	DISS	STATION	CNR	GAZ
11834	MANNING	STATION	CNR	GAZ

51.Getarp name source of hill for high = hilly

relation is HILL
relation is HIGH
relation is HILLY
The required path of hill in the hierarchy is :

HILL HILLY HIGH TERRAIN ROOT

BADGER HILL GAZ
PORCUPINE HILLS GAZ
ANTLER HILL GAZ
DRIEDMEAT HILL GAZ
BUFFALO HILL GAZ

52.Getarp name

ERROR in command specification
Correct usage:
Getarp attribute(s) of relation for relation1=relation2

53. Getarp name of hill for high = xyz

relation is HILL
relation is HIGH
No relation matches specified attribute xyz

54.Find name

Syntax error in command usage. Too few parameters.

55. Find name of xyz where name EQ HILL

No relation matches specified attribute xyz

56. Find name of river where source EQ GAZ

relation is RIVER
relation is RIVER2
More than one relation matches specified attribute river

57.Find name feature source of Indian.reserves where lat3 EQ 0

relation is INDRES

PUSKIAKIWENIN 122	INDIAN RESERVE	GAZ
BLOOD 148	INDIAN RESERVE	GAZ
BLACKFOOT 146	INDIAN RESERVE	GAZ
EDEN VALLEY 216	INDIAN RESERVE	GAZ
FREEMAN 150B	INDIAN RESERVE	GAZ
NAMUR RIVER 174A	INDIAN RESERVE	GAZ

BLANK PAGE
PAGE EN BLANC